Improving Reliability Of Wireless Sensor Networks for Target Tracking using Wireless Acoustic Sensors

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Raghu Kisore Neelisetti

Certificate of Approval:

Improving Reliability Of Wireless Sensor Networks for Target Tracking using Wireless Acoustic Sensors

Raghu Kisore Neelisetti

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
December 18, 2009

Improving Reliability Of Wireless Sensor Networks for Target Tracking using Wireless Acoustic Sensors

Raghu Kisore Neelisetti

Signature of Author

Date of Graduation

Vita

Raghu Kisore Neelisetti, son of N.Hanumantha Rao and N.Rama Devi, was born May 16, 1979, in Tenali, India. He earned his Bachelor's degree in Electronics and Communication Engineering from Sri Venkateswara University, Tirupati, India in 2000. In 2002, he completed his Master's degree in Computer Science at Indian Institute of Technology-Madras, India. After his Master's he worked for nearly 2.5 years as research staff on Network Management Systems for Central Research Laboratory, Gaziabad, India.

Dissertation Abstract

Improving Reliability Of Wireless Sensor Networks for Target Tracking using Wireless Acoustic Sensors

Raghu Kisore Neelisetti

Doctor of Philosophy, December 18, 2009
(M.S., Indian Institute of Technology-Madras, 2001)

159 Typed Pages

Directed by Alvin S. Lim

The advancement of MEMS technologies has made it possible to produce tiny wireless sensor devices. These tiny sensors hold the promise of revolutionizing sensing in a wide range of application domains because of their flexibility and low cost. One such application is target localization and tracking using acoustic signal of the target. The capabilities of these tiny devices are limited by their battery power, storage capacity, computational power and communication bandwidth. These limited capabilities make the decisions made by each sensor error prone. Hence most target detection and tracking algorithms require the sensors to work in groups in order to improve the reliability of target tracking algorithms. This makes it necessary for deployed sensors to discover and group together so that their coverage can be maximized. In addition, with the advent of video sensor networks it has become possible to record a video of the target once it is detected and later be relayed to an external agent. In this paper, we propose a clustering algorithm that tries to produce the optimal number of possible clusters for any sensor deployment scenario. The proposed clustering algorithm is distributed in nature and has the ability to reconfigure in the event of node failure. The algorithm is localized in nature and hence does not need flooding across the entire network. Since the algorithm allows for more than one cluster to track the same region the system reliability is greatly improved. The algorithm achieves 97% coverage for all the node deployment scenarios evaluated. In each case the average probability of detection achieved is 92% of the theoritical best possible. The other metrics evaluated are support weight and breach weight. The clustering algorithm achieves 89% and 91% of the theoritical best possible. In each of these cases the algorithm is able to form clusters in about 5 seconds of the simulation time.

On successful detection of a target, the video information needs to be relayed to an external agent which could be several hops away from the point of detection. The lossy nature of wireless links makes the end-to-end delivery ratio decrease exponentially. We address the problem of end-to-end reliability by proposing reliable directed diffusion (RDD) that uses a localized route repair algorithm that does not require a global re-flooding. A route repair algorithm is important to directed diffusion (DD) as the path selected by the protocol is not based on any historical data of link quality and hence prone to packet losses. The node density and power constraints of sensor networks coupled with the ever changing link quality makes it difficult for a node to keep track of its links and hence choose the best possible path. We present the design and implementation of the reliable directed diffusion. RDD repairs the established paths locally by using backup nodes, i.e. nodes that can overhear the positive reinforcement and the corresponding data packets that go in the reverse direction. RDD detects failures at the sender through the MAC layer. Reliable Directed Diffusion provides 30% improvement in the delivery ratio. The end-to-end delay is only 3% more than that of Directed Diffusion. Finally, the average energy consumed is only 5% more than that of Directed Diffusion.

Acknowledgments

First, and foremost, I would like to thank my advisor Dr. Alvin Lim for having accepted me as a Ph.D. student. I will always be indebted for all the support, guidance, encouragement, and understanding he has given me throughout my research. Particularly, his suggestions, criticism, guidance and general exchange of ideas contributed so much to this dissertation. His mentor ship is paramount in providing a well rounded experience, which will be valuable treasure in future career. I would also like to express my gratitude to the advisory committee members, Dr. David Umphress, Dr. Drew Hamilton and Dr.Levent Yilmaz.

I would like to thank Dr.David Umphress for the support he has provided as Graduate programme Officer and making my stay at Auburn a memorable one. I also thank Dr. Drew Hamilton for his valuable courses (especially simulation of computer networks) that have helped me better understand the finer aspects and limitations of event based simulators.

I thank Dr.Prathima Agrawal, Ginn Distinguished Professor and Director of Wireless Engineering Research and Education Center, for introducing me to the field of wireless networks and for her constant feedback from time to time. I would like to thank her for her valuable support through the Vodafone Fellowship without which this work would have been impossible. I have thoroughly enjoyed her seminar series which has become a venue for the students to exchange ideas and has given the opportunity to listen to the best professors from other institutes. I finally appreciate her role as an out side reader of this dissertation and the valuable suggestions she has provided.

I thank all the great colleagues I had the pleasure to meet at the Mobile Computing Lab. I would also like to thank my parents for their love, inspiration, guidance, and encouragement. My friends provide me with everything that is required to reach my dreams. They have been there always sharing my happiness and extending their hands when going was not easy. I am deeply thankful to my friends without whom I can not imagine myself. I am blessed to extend them my sincere and heartily gratitude for providing me with selfless motivation, courage, confidence, and support.

Finally, I would like to thank the Vodafone fellowship programme for funding the last couple of years of my research.

Style manual or journal used Journal of Approximation Theory (together with the style known as “aums”). Bibliograpy follows van Leunen’s *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The increasing capabilities and declining cost of computing and communication devices, has led to an increase in the number of applications of wireless sensor networks. Some of the these applications are earthquake monitoring, environmental monitoring, home and office controls, medicine and security, etc. In each of these applications the wireless nodes sense physical characteristics of the world. The physical characteristics include temperature, acoustics, light and pollution. Each sensor on sensing the physical characteristics needs to deliver the information to base stations or external agents. These external agents could be about 20-30 hops away depending on the size of the sensor field. The task of using sensors in sense and response systems is complicated by their limited resources. The individual sensors are constrained by their limited processing power, short range communication and a small amount of storage. The other factors that affect the performance of Wireless Sensor Networks (WSN) in real world are fault tolerance, scalability and topology change. Further when compared to ad hoc networks, sensor networks are several magnitudes larger and at times several thousand nodes could form one single large co-operative network. Over years, several routing protocols and route repair mechanisms have been proposed for ad hoc networks, but none of these are scalable to networks of the order of several thousand nodes. This is largely because sensors have limited memory and limited power and hence not possible to gather and store routing information for a large network.

In this work we look at another interesting application, battlefield surveillance. Surveillance involves both detection and tracking of intruders. In general, target classification and tracking algorithms rely on information provided by a cluster of sensors. In case of target classification each sensor is equipped with different modalities, such as magnetic, radar, thermal, acoustic, chemical, electric, seismic and optical. Hence the target classification draws its results from observations made by a cluster of modalities. This emphasizes the need for a clustering algorithm that can exploit the redundancy in the sensor deployment and reduce the latency in the exchange of raw data and the amount of raw data that needs to be exchanged. Tracking, based on the strength of acoustic signal received by a set of sensors is another common technique. This also requires deployed sensors to work in groups. Though each sensor is capable of detecting the presence of a target, its results are error prone and could result in false alarms. Hence to increase the accuracy of the detection algorithm it becomes necessary to fuse the measurements of a group of sensors. This makes it necessary for the deployed sensors to work in small clusters, so that the overall reliability of the surveillance system can be improved.

Wireless sensors can be used to detect various target features such as thermal signatures, ferro-magnetic content or acoustic signal. The absence or presence of a target phenomenon can be inferred by aggregating the measured values from a small group of sensors deployed. In this work we assume that each sensor is equipped with a microphone and hence can record the acoustic signal.

Detection requires that the system discriminate between a target's absence and presence. Successful detection requires a node to correctly estimate a target's presence while avoiding false detections in which no target is present. This can be done by using triangulation method based on acoustic measurements made by at least three sensors. On the other hand target tracking is more complicated since it involves maintaining the target's position as it moves over time in a region covered by the sensor network's field of view. Therefore the tracking algorithm should be able to identify the orientation of the target's path and its velocity in addition to geographical location of the target. One such algorithm to track the target is the CPA (Closest Point of Approach) algorithm [1]. In this algorithm a group of four sensors make's CPA measurements, and then, based on the measurements, the trajectory of the target can be deduced with reasonable accuracy using CPA algorithm. The capability of the application can be further enhanced if we assume that each sensor is equipped with a video camera. This is possible with the advent of low cost video cameras that are capable of providing resolution in the order of mega pixel. The CPA measurements made by each of the four sensors will be reported to a node (actually one of the four sensors) where the target's trajectory can be computed. Once the parameters of the target's trajectory are computed, the camera associated with the sensor can be programmed to pan in the target's direction. However, this requires real time data from each of the four individual sensors, i.e. the CPA measurements generated by the sensors must be delivered to a node in a timely fashion. Out-dated reports are of little use. These timing constraints call for a robust clustering algorithm.

The overall system architecture consists of two self contained components: the acoustic target tracking subsystem which deals with the detection and processing of acoustic signals and the communication subsystem which is responsible for exchanging sensor data and high quality tracking results. One way to address the limited computational and battery power of wireless sensor devices is to organize the sensors into clusters. Sensors in each cluster coordinate in sensing and communication to perform the sensing task. To deal with the inaccuracy in measurement and unreliability typical of low end devices in remote or hostile environments, we suggest a clustering algorithm that organizes the sensors into redundant or overlapping clusters so as to obtain more robust results. The proposed clustering algorithm is distributed in nature and the number of clusters to be formed can be easily controlled. Further, since the cluster head chooses its member nodes from its one hop neighbors, the raw data has to travel only one hop. Finally, the target tracking results of each cluster head can be progressively fused with those of it's neighboring clusters.

Sensor networks are traditionally meant for applications where high loss rates are acceptable. For example, average temperature measurements from a sector of a forest which will not to vary drastically. However, sensor networks are becoming more prevalent in complex and sophisticated application domains, such as sense and response systems, critical infrastructure, and industrial and manufacturing control systems where high packet losses are unacceptable. Specific examples include real-time target tracking systems and tornado and tsunami detection networks. Moreover, emerging sensor networks will support streaming audio and video applications. For example, in the above mentioned target tracking system, the recorded video needs to be delivered to the external agent that could be several hops away. Because of the stringent requirements of these systems, end-to-end reliability is of paramount importance. Providing end-to-end reliability is complicated because of the resource constraints and the nature of losses. The communication medium in sensor networks is wireless which is more error-prone than wired networks. Packets in sensor networks are lost due to link failure (caused by interference, radio noise, etc.), node failure or congestion. The very nature of the wireless medium makes link failures temporary, though more common. In unknown terrain, the behavior of wireless links may be highly unpredictable and have higher error rates than expected. Additionally, the harsh environments in which sensor networks are deployed result in node failures which are less frequent than wireless errors but permanent.

Data-centric routing protocols, such as directed diffusion [2], have been commonly used for wireless sensor networks because of their energy efficiency and scalability. Directed diffusion enables sensor data to be disseminated from data sources to sinks with low delay. Our approach to providing reliability in directed diffusion is to efficiently and locally repair the route when there are link or node failures. The first step in route repair is to identify route failure. This is important because sensor networks are event-based and the events may not occur periodically. Hence, in the case of low event rate, route failures cannot be identified based only on the absence of the events in the network. Secondly, it is very important to differentiate between temporary and permanent failures. If the route repair strategy is too aggressive, the network might misinterpret more frequent temporary failures as permanent failures and spend more time and resources in unnecessary route repair. On the other hand, a less aggressive strategy would treat permanent node failures as temporary failures causing packets from the source to the sink to be lost. Finally, route repair is difficult in directed diffusion because of the data-centric nature of the protocol. In directed diffusion, routing is always done based on attribute/value pairs so the absence of source/destination information makes it hard to discover an alternate path. Therefore, the route repair mechanism has to rely on attribute/value pairs rather than just ID.

In this work we present an improved version of directed diffusion, a protocol that can repair both temporary and permanent path failures. Reliable Directed Diffusion (RDD) makes use of the fact that nodes which are geographically closer to the sender experience better link quality than the nodes which are farther away. The results show that reliable diffusion improves delivery ratio by about 20% in the case of a grid topology and about 35% in a random topology. These improvements are achieved without any significant increase in energy consumed or end-to-end delay. Furthermore, RDD has been shown to be a highly scalable protocol given that its overhead increases as a linear function of the path length. RDD significantly improves the reliability of directed diffusion for both permanent node failures and temporary link failures without incurring costly overhead. RDD make substantial contributions to directed diffusion and to data-centric routing.

Chapter 2 gives background information about directed diffusion and an overview of research related to target tracking systems and in Chapter 3, we discuss the motivations for our work by first briefly explaining the target tracking system and the need for further improvements. We describe the architecture of the proposed system in Chapter 4 and some preliminary results in Chapter 5. We conclude the proposal with a summary of the tasks to be accomplished along with a schedule of activities.

Chapter 2
Motivations, Objectives, and Applications

Individually tiny sensors may appear to be of little value but a wide range of applications arise when deployed in large scale co-operative networks. One such application is target detection and tracking in a hostile environment such as a battlefield. In this application we are not only concerned with target detection but also tracking of the target. Though there are quite many algorithms in literature that deal with reliable detection of targets, there is only one algorithm that best suits reliable tracking of the target and it is Closest Point of Approach or CPA algorithm.

Real-time tracking of moving targets using wireless sensor networks has been a challenging problem because of the high velocity of the targets and limited resources of the sensors. CPA (closest point of approach) algorithms are appropriate for tracking fast-moving targets since the tracking error is roughly inversely proportional to the square root of the target velocity.

In this chapter we first briefly explain the target tracking system we developed at Auburn University and then explain the need for improvements.

2.1 Closest Point Approach Algorithm.

CPA algorithm is a data fusion algorithm that relies on the raw data gathered by individual sensors to successfully estimate the target parameters like velocity, orientation of the target's path in addition to the location of the target. Each sensor monitors the acoustic signal from the target with the help of a microphone, i.e. it monitors the signal energy for a given time window. The sensor confirms the presence of a target (called event) once the signal strength exceeds a certain threshold. The threshold is dynamically updated based on background noise statistics to reduce false alarm rate. Once a node detects an event (i.e. the presence of a moving vehicle), it stores a time series segment corresponding to the event. Figure 2.1 shows the time series segment corresponding to the interval in which the energy first exceeds the threshold (start of event) and eventually drops below the threshold (end of event) after reaching a peak value. The time at which the acoustic signal peaks is called the closest point of approach (CPA). The CPA measurements made by each of the four sensor are reported to a centralized location (which is just any of the four sensors) where the individual measurements are fused together by the CPA algorithm to determine the target motion parameters, such as precise location at a certain instant of time, velocity and orientation. Once these parameters have been calculated, the camera attached to the sensor can be programmed to record a video of the target and the recorded video can be sent to an external actor by using a data centric routing protocol, such as Directed Diffusion.

2.2 Features of Current Target Tracking System

We designed and developed an integrated system for target detection, tracking and image/video capture of moving targets using collaborative mixed wireless sensor nodes. This system integrates and interoperates the algorithm for accurately computing target location, velocity and trajectory direction with the other communication and control software for controlling camera sensor nodes for image/video capture of target at its predicted location. The wireless ad-hoc sensor network uses a different network protocol, i.e. directed diffusion,

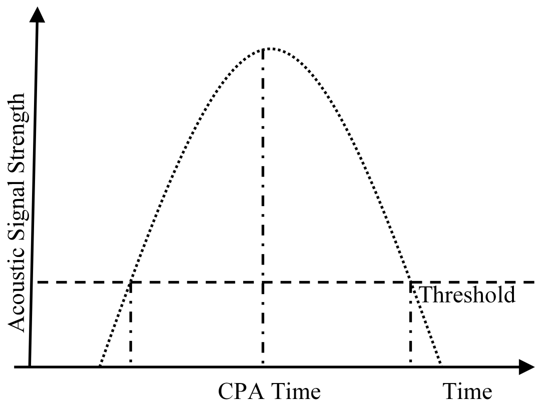

Figure 2.1: Event detection by thresholding the energy of the acoustic signal detected by the microphone. The horizontal line represents the threshold. The maximum reading corresponds to CPA time.

whereas the camera sensor node uses TCP/IP network protocol. We describe the details of the architecture, design and implementation of the target detection and tracking system for detecting, locating and identifying of moving targets. Sensors detect and analyze acoustic data to determine the closest point of approach (CPA) time of the target. Intuitively, CPA is the time when the target signal is highest and therefore was closest to the sensor node. Sensors send CPA time information the task group leader, the cluster head, for analysis. The cluster head uses the CPA algorithm described in [1] to predict the position, velocity, and direction of the target. This information is sent to a camera control application on an IP network through a bridge node that interconnects the directed diffusion and IP networks. The camera control node pans and zoom a camera to view the target's current location. Video and/or images of the target may then be captured by a video capture application for further analysis. These systems were tested in several field experiments and the results and performance evaluations are included in this report. In particular, we present some results of the target tracking experiments and show the clips from one of the videos of the target captured by the camera at the predicted location. We also conducted several tests to ensure that the system is robust with regards to the sensor networking, target detection methods, CPA computation and the camera control algorithms. We will also develop more advanced methods to ensure robustness in the sensor network, mixed network and sensor detection system. As we have demonstrated the proof of concept, i.e. capability to detect and predict location and velocity of target and capturing its image/video, we have begun to focus on other more practical issues that will make this system more applicable to realistic scenarios.

2.3 Motivations

2.3.1 Reliable Detection

In this section we present some of the shortcomings of the existing system and briefly explain how the proposed clustering algorithm solves them.

The CPA algorithm used in the Current system has pitfalls. There are certain configurations in which the algorithm fails to estimate the motion parameters. Figure 2.2 shows all the possible target trajectories with respect to the way sensors are deployed. The CPA algorithm can estimate the target parameters only when there are uneven number of sensors deployed on either side of the targets trajectory. At the minimum, to detect the target motion parameters we need CPA measurements from four sensors with 3 on one side and one on the other side of the target trajectory. Also the algorithm requires the three sensors that are on one side of the trajectory to be non collinear. In all other cases the solution is ambiguous [1].

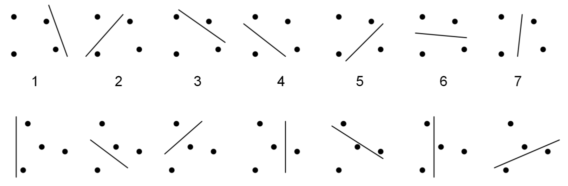

Figure 2.2: Classification of target trajectories according to the way of sensor field decomposition.

The major disadvantages of the current system can be summarized as:

- The CPA algorithm in the current system requires CPA measurements from at least four nodes and the four nodes need to be unevenly deployed with respect to target's trajectory. Since, in the Real world we do not control the target's trajectory we cannot assure uneven deployment of the four sensors.

- The video captured by the video camera needs to delivered to an external agent over a multi hop wireless network. The current system uses directed diffusion and directed diffusion is ill equipped for video sensor networks.

2.3.2 Improving Detection Reliability with Redundant Node Clusters

The clustering algorithm proposed in this work avoids the issue of even deployment of sensors by grouping five sensors together into one cluster. One of the five sensors is chosen as the cluster head. All the member nodes send their raw data to the cluster head, which then makes use of the CPA algorithm to estimate the target motion parameters. Figure 2.3

shows a cluster formed between five nodes and the dotted lines represent the possible paths a

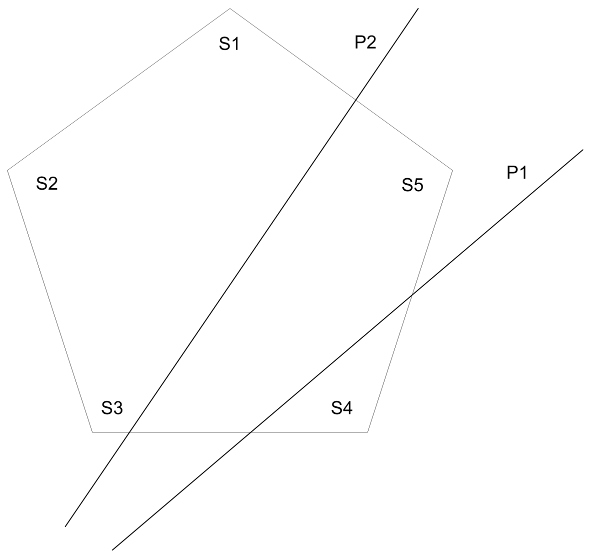

Figure 2.3: Classification of target trajectories according to the way of sensor field decomposition.

target can take. Each cluster will have a cluster head to which all the other sensors can send in their results. The cluster head then runs the CPA algorithm to identify the trajectorys parameters. The cluster head needs CPA measurements from only 4 sensors, but does not know which 4 measurements will lead to a solution and hence tries out combinations. It then ignores the invalid combinations and averages the valid solutions.

2.3.3 Reliable Communication

Wireless sensor networks are originally designed as distributed event-based systems that differ from traditional communication networks in several ways. These networks typically have nodes with severe energy constraints, variable quality links, low data-rate and many-to-one event-to-sink flows. Recently, Wireless Multimedia Sensor Networks (WMSNs) have been developed due to the availability of lowcost cameras, microphones, and other sensors producing multimedia data. The applications, accordingly, are extended to video surveillance and notification, video and computer assistance in video-assisted living and healthcare. The stringent requirements of real-time multimedia applications include end-toend delay, bandwidth and loss during data transmission. Communication algorithms for WMSN must therefore be specially designed to operate efficiently under these constraints. Directed diffusion is a datacentric protocol designed for wireless sensor networks. However, it is not efficient in more challenging domains, such as video sensor networks, because of its inability to satisfy the throughput and delay requirements of multimedia data.

In the current target tracking application the video camera associated with individual sensor starts to record the video of the target once it appears. The recorded video needs to be transferred to an external agent that could be about 20-30 hops away, depending on the size of the sensor field. Though directed diffusion is a preferred routing protocol for wireless sensor networks, but is ill equipped for video sensor networks. We propose Reliable Directed Diffusion that improves end to end reliability by recovering losses on a per hop basis.

2.4 Objectives

2.4.1 Reliable Detection

The target detection algorithm is prone to errors. The primary source of errors is detection of CPA time by the sensor. The ambient noise recorded by the microphone along with the target's acoustic signal makes it difficult to configure the right threshold for the acoustic signal strength. Setting the threshold to a low value leads to false alarms while a value that is too high might lead to the sensor not being able to record the CPA event. The second source of errors are node failures. Once a node fails the CPA measurements made by other nodes in the cluster may become useless.

The proposed clustering algorithm improves the overall system reliability by forming a series of overlapping clusters. This would mean that there would be more than one cluster that would track the same area. Such redundancy would improve the overall system reliability and make it less prone to errors.

Each sensor uses heuristics and forms only an optimum subset of all the possible polygons. The proposed clustering algorithm has several advantages:

- is distributed in nature and the number of clusters to be formed can be easily controlled.
- a series of overlapping clusters improves system reliability and allows for tracking of curvilinear trajectories.
- since the cluster head chooses its member nodes from its one hop neighbors, the raw data has to travel only one hop.
- finally, the target tracking results of each cluster head can be progressively fused with those of its neighboring clusters.
- a series of overlapping clusters allows for tracking of curvilinear targets.

2.4.2 Reliable Communication

The basic idea of Reliable Directed Diffusion is that the link quality degrades with distance. This is a reasonable justification and this characteristic is clearly demonstrated by Figure 2.4. It can be seen that the link changes erratically as the distance between sender and receiver changes from 20m to 30m. Hence in the event of an unsuccessful transmission it is better to deliver the packet to a node that is geographically closer and this node would in turn relay the packet to the intended receiver. we use the relay nodes only when the primary hop turns bad. This way we can optimize the number of transmissions and receptions needed to deliver data and at the same time achieve higher reliability. The data in Figure 2.4 is from [3].

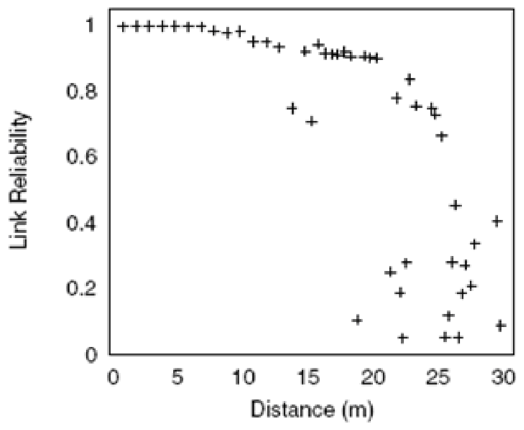

Figure 2.4: Link Quality Vs. Distance Profile

Directed Diffusion chooses the best path based on end-to-end delay. Our goal is to continue using this low delay path identified by directed diffusion as long as we can. In the event of link failures the idea is to reroute the data packets through an intermediate node (also called the back up node) that is in between the sending and receiving node and hence closer to both the nodes. Since the backup node is closer to both the sender and receiver the link quality is better than the link between actual sender and receiver. Finally, we gain the advantage of low end-to-end delay by using the path identified by Reliable Directed Diffusion and also improve delivery ratio by rerouting the failed packets through shorter and better links in the event of link failures.

2.5 Applications

Recent developments in sensor techniques make wireless sensor networks (WSNs) available to many application domains. Some of these applications are battlefield surveillance, disaster and emergency response. The current research in Wireless Sensor Networks (WSN) is widespread and pervasive in many disciplines because of the potential to embed tiny, inexpensive, low-power sensors in many environments to provide a wide range of surveillance and monitoring applications. A key advantage of WSN is that the network can be deployed on the fly and can operate unattended, without the need for any pre-existing infrastructure and with little maintenance. Typically, sensor nodes are deployed randomly (e.g., via aerial deployment), and are expected to self-organize to form a multi-hop network. A sensor node is capable of sensing some physical phenomenon (e.g., detect tank vibrations or sniper gun noise), processing the sensed data and communicating the observed measurements to fusion nodes, also called micro-servers. The sensor nodes may also perform data aggregation/compression to reduce the communication overhead in the network. In this paper, we investigate the design trade offs for using WSN for implementing a system, which is capable of detecting and tracking military targets such as tanks and vehicles. Such a system has the potential to reduce the casualties incurred in surveillance of hostile environments.

Chapter 3

Related Work

In this section we first evaluate different routing protocols proposed for MANETs and bring out the disadvantages of using them for WSNs. We then explain how directed diffusion overcomes those disadvantages.

3.1 Unicast Routing Protocols for MANETs

Existing routing protocols can be classified either as proactive or reactive. Proactive protocols attempt to continuously evaluate all of the routes within a network so that when a packet needs to be forwarded, a route is already known and can be used immediately. OSPF is an example of a Proactive Routing Protocol (PRP) for wired IP backbone networks. MANET-specific examples include Optimized Link State Routing (OLSR) [4], Topology Broadcast based on Reverse Path Forwarding (TBRPF) [5] and Hazy Sighted Link State Routing [6]. In contrast, Reactive Routing Protocols (RRPs) invoke a route determination procedure on-demand only. Thus, if route is needed then some sort of global-search procedure is employed. The classical flood-search algorithms are simple reactive protocols. MANET-optimized examples include Ad hoc On-Demand Distance Vector (AODV) [7] and Dynamic Source Routing (DSR) [8].

It is well-known that proactive protocols are not optimal for MANETs that have rapidly changing topologies. However, purely reactive protocols are often inappropriate for several common MANET topologies such as cluster-based networks and relatively static networks.

In addition, reactive protocols introduce additional latency (and possibly overhead) for real-time traffic. As such, hybrid or zone routing protocols that use a mix of both proactive and reactive routing techniques at each network node have been proposed. One example is Cornells Zone Routing Protocol (ZRP) [9]. The above protocols are unicast in nature and hence do not serve the purpose of multiple sources and sinks communicating with each other as is the case in sensor networks. Further as the size of network grows so does the amount of routing information that needs to be stored by each node. The limited memory of sensor motes makes this difficult.

3.2 Multicast Routing Protocols for MANETs

A number of ad hoc network multicast routing protocols have been proposed over the past few years as well [10],[11],[12], [13], [14],[15],[16], [17],[18], [19], using a variety of basic routing algorithms and techniques. Of these multicast routing protocols, a few attempt to operate in an on-demand fashion [10],[13],[14], [15], [20], in which the operation of the protocol is driven by the presence of data packets being sent rather than by continuous or periodic background activity of the protocol. Most of the multicast routing protocols mentioned above include both proactive and reactive mechanisms. Routing functionality can roughly be divided into two parts: route discovery and route maintenance. Most multicast routing protocols perform the route discovery part on-demand and most perform the route maintenance part proactively, e.g. they use periodic neighbor detection packets, or refresh the multicast state periodically.

For example, the On-Demand Multicast Routing Protocol (ODMRP) [10] builds multicast meshes through periodic network-wide control packet floods. The protocol relies on these floods to repair link breaks in the mesh that occur between the floods. The Multicast Ad Hoc On-Demand DistanceVector protocol (MAODV) [21] requires continuous periodic neighbor sensing for link break detection, and periodic group hello messages for multicast forwarding state creation. The hello messages are sent regardless of whether or not there are any senders for the multicast group in the network, as long as there is at least one receiver. Similarly to MAODV, the Associativity-Based Multicast (ABAM) protocol [12] requires continuous periodic neighbor sensing for link break detection and distribution of link characteristics. In addition, these protocols rely on explicit prune messages for deletion of forwarding state that is no longer needed. Loss of an explicit prune message because of wireless interference or because the sender of the prune message has moved out of range of the intended recipient of the prune, leads to significant unnecessary overhead as nodes continue forwarding packets even though there are no receivers for the group that are interested in receiving them downstream. The protocol overhead incurred to maintain the multicast meshes is a bottleneck for their use in WSN as the sensor motes are heavily energy constrained.

Though the above protocols provide for multicast communication, they involve considerable protocol overhead to maintain multicast information and also need space to store the multicast routing information.

3.3 Directed Diffusion

Directed Diffusion [2] is data centric in that all communication is for named data. It further assumes that all the nodes in a WSN are application aware. Data generated by sensor nodes is named by attribute-value pairs. Sinks are the nodes that are looking for specific information in the network. The interested nodes request by sending interests for named data. Data matching the interest is then drawn toward the sink nodes. Intermediate nodes can cache, or transform data, and may direct interests based on previously cached data. The data centric approach of Directed Diffusion makes it significantly different from IP based routing protocols. In an IP based routing protocols nodes are identified by their end points and inter-node communication is layered on an end-to-end delivery service provided within the network.

3.3.1 Protocol Overview of Directed Diffusion

The Protocol has four types of messages to establish paths in a network. Early work on Directed Diffusion [2] described the basic concept of diffusion based on these messages and this basic approach is now called two-phase pull diffusion. Though the basic approach is ideal for some applications, it has been found to be a poor match for other classes of applications. Hence, as experience with sensor applications grew, a family of algorithms have been built from the basic primitives. Today, in addition to the basic approach directed diffusion also operates in two additional modes: one phase push, and one phase pull. In this work, we consider the conventional two phase pull communication model. The two phase pull mechanism makes use of all the four types of messages to establish paths within the network. A sink node subscribes to a data flow by flooding the network with interest messages that name the type of data the sink wants to receive. Intermediate nodes store the interest and record the neighbor from which it was sent. This saved path leading to the sink is known as a gradient. The state of the gradient is set to false indicating that it will not forward data packets until it is later enabled by a positive reinforcement message. Source nodes with data matching the interest publish exploratory data along the gradients previously created. Upon receiving exploratory data, the sink sending a positive reinforcement message to its single fastest neighbor (i.e., the neighbor that delivered the first exploratory data message). Any node that receives a positive reinforcement message will, in turn, reinforce its fastest upstream neighbor until a reinforced path all the way back to the source is established. Slower paths may be negatively reinforced. Subsequent data emanating from the source, known as reinforced data, will be unicast or multicast over the reinforced path to the sink(s). This two-phase process results in the creation of a multipoint-to-multipoint distribution tree.

3.3.2 Advantages of Directed Diffusion

Directed diffusion has generally been proven to be well-suited to sensor networks. Its primary advantages are data-centricity, reactive nature, aggregation and multipoint-to-multipoint links. Its data centric model is more appropriate for many sensor network applications. It performs more efficiently and provides more useful services for many types of sensor network applications (e.g. query processing). The use of named data provides an energy efficient mechanism for routing data, which avoids the unnecessary complexity of host information. Since data is the primary concern it makes sense to use it as the primary routing criterion instead of host address, a property largely unimportant in sensor networks. To deal with broken paths, diffusion periodically re-creates routes by performing the same procedure used to initially find routes: global flooding. Since diffusion handles failure and mobility with global repair mechanisms, the costs incurred for repair are significant. To reduce the energy costs of global repair, the path maintenance mechanism is performed on a relatively infrequent schedule (e.g. every 60 seconds). In the worst case, data may be delayed an entire refresh interval before a broken path is repaired.

3.3.3 Disadvantages of Directed Diffusion

The data centric approach of Directed Diffusion makes it scalable for routing packets in large wireless networks and also provides for a multipoint to multipoint communication. In spite of these advantages, it has inherent drawbacks.

Directed Diffusion makes use of end-to-end delay to determine the best path between the source and the sink. Since the end-to-end delay is not based on historical information, it does not necessarily represent a stable path. This makes the path more prone to wireless losses. Further Directed Diffusion does not have a proactive way of determining and recovering from losses. While this may be adequate for some applications, it may be completely unacceptable for others, e.g. time-critical and real-time systems. Standard directed diffusion includes two mechanisms for path repair. Diffusion was designed to handle path failure primarily by the periodic re-creation of gradients using a global mechanism. The designers of diffusion also mention a local repair procedure, but fail to adequately deal with the route repair problem.

3.4 Existing Solutions for providing end-to-end reliability

One way to avoid wireless losses is by choosing a better technique for selecting the path. Ad hoc network protocols use link quality as the metric for choosing the next hop. Determining link quality is done either through the Protocol Model or the Physical Model. The Physical Model determines link quality based on received signal strength. Although inexpensive, the Physical Model may not be accurate since the wireless link quality continually changes. On the other hand, the Protocol Model makes use of probe packets to estimate the wireless link quality. In the context of sensor networks, the use of probe packets is expensive and hence may not be appropriate for power-constrained sensor nodes. Link quality estimation is further compounded by the high density of sensor networks. Because of higher node density each node may have a large number of neighbors and proportionately more links whose quality has to be estimated.

In sensor networks, packet losses result from wireless losses and network congestion. Network congestion leads to unnecessary retransmissions that waste network bandwidth and reduce the lifetime of the sensor nodes. In the literature, many techniques have been suggested to improve end-to-end reliability in sensor networks. These techniques are mostly transport layer protocols.

3.4.1 PSFQ

Pump Slowly Fetch Quickly (PSFQ) [22] is proposed for reliable retasking/reprogramming in WSNs. PSFQ slowly injects packets into the network, while performing aggressive hop-by-hop recovery in case of packet losses. The pump operation in PSFQ simply performs controlled flooding and requires each intermediate node to create and maintain a data cache to be used for local loss recovery for packet loss and in-sequence data delivery. Although this is an important transport layer solution for WSNs, it is applicable only for strict sensor-to-sensor reliability and for purposes of control and management in the direction from the sink to the sensor nodes.

3.4.2 ESRT

Event-to-Sink Reliable Transport (ESRT) [23] is based on an event-to-sink reliability model and provides reliable event detection without any intermediate caching requirements. Though it seeks to achieve minimum energy expenditure and has the congestion control component, it fails to address the issue of packet losses due to link and node failures. In low event rate applications, the primary causes for packet losses are link and node failures rather than congestion.

3.4.3 RMST

In [24], the authors propose a Reliable Multi Segment Transport (RMST) layer for directed diffusion. They evaluate the placement of reliability for data transport at different levels of the protocol stack. The authors conclude that reliability is important not just to provide hop-by-hop recovery for the transport layer, but also because it is needed for route discovery and maintenance. In RMST, receivers are responsible for detecting whether or not a fragment needs to be re-sent. The performance of the protocol depends on the caching policy and further the receivers depend on NACK mechanism to inform the senders about the packet failures. RMST is the closest protocol to the proposed Reliable Directed Diffusion and we evaluate the performance of RMST and Reliable Directed Diffusion and show that our idea performs significantly better than RMST.

3.4.4 CBQ

A cluster based forwarding (CBF) protocol is proposed in [25]. In CBF, each node forms a cluster such that any node in the next-hops cluster can take forwarding responsibility. CBF is not a routing protocol, but rather is designed as an extension layer that can augment existing routing protocols. Each node selects a subset of its neighbors as its helpers using its helper admission algorithm. The helper admission algorithm requires each node to gather link quality information from each neighbor by exchanging sequence-number-stamped packets. The node then broadcasts the gathered information to its neighbors. The forwarding scheme is based on allocating time slots to the helper nodes which may add additional overhead to the sensor network.

3.4.5 CODA

In [26], the authors propose an energy efficient congestion control scheme for sensor networks called CODA (Congestion Detection and Avoidance) which comprises three mechanisms. The first mechanism helps to identify network congestion. Secondly, CODA uses a combination of the present and past channel load conditions and the current buffer occupancy to infer accurate detection of congestion at each receiver with low cost. Finally, once congestion has been detected, CODA uses either a hop-by-hop back pressure (open loop) or multi source regulation (closed loop) to alleviate congestion. The above techniques help to improve the end-to-end reliability by reducing packet losses due to congestion. They do not address the issue of wireless losses however. In fact, they do not differentiate between the wireless and congestion losses. In contrast, reliable directed diffusion helps to reroute packets in the presence of wireless losses and also fix broken routes caused by node failures.

3.4.6 Other Protocols

Several routing protocols have been proposed for MANETs which have a provision for route repair [27] [28] [29] [30] [31]. Similar protocols cannot be used for sensor networks, however, because of their limited scalability. These protocols need cache to store routing information which can grow in size as the network size increases. Sensor networks are built with low-powered devices that have limited memory. Hence, a data-centric protocol, such as directed diffusion, is preferable. Two techniques that come closest to our work are Witness Aided Routing (WAR) [32] and ASCENT [33]. WAR makes use of witness nodes to reroute failed packets. Once again, WAR is a solution for MANETs. In ASCENT, the authors introduce a new layer between the network layer and the link layer which makes the protocol independent of the network layer routing protocol. In the event of failure, the sender tries to route the packet through a node in the vicinity of the intended receiver, called a passive neighbor. The disadvantages are the high number of configurable parameters that must be fine tuned and the high communication overhead in identifying passive neighbors. These nodes are identified by introducing two new messages: neighbor announcement messages and help messages. These two packets must be sent every time the passive nodes are to be made active. This process could be very expensive in sensor networks since the packet losses are more frequent. Thus the recovery mechanism proposed by ASCENT is both time and energy consuming.

3.5 Target Tracking Systems

The system that we are presently referring to is an intrusion detection system which is essentially a surveillance situation of practical importance and is well-suited to wireless sensor networks. The intrusion detection system is designed as a dense, distributed, wireless network of multi-modal, resource poor sensors combined into loosely coherent sensors that perform in situ detection and estimation. There are several issues of interest in designing such distributed intrusion detection systems. First and foremost is the sensor deployment algorithms. These algorithms aim at maximizing the field of coverage of a given set of sensors. One metric to identify the effectiveness of a deployment strategy is by measuring the worst and best case coverage paths. In [34] the authors optimize deployment of heterogeneous sensors through Linear Programming. In [35] the authors propose three approximation algorithms for a variation of the SET K-COVER problem, where the objective is to partition the sensors into covers such that the number of covers that include an area, summed over all areas, is maximized. In [36] the authors analyze the minimum number of nodes needed for random deployment so as to meet a desired value for least path of exposure metric. They assume Gaussian distribution for the random deployment strategy. In [37] the authors propose algorithms to provide k-coverage in a mostly sleeping network. The aim of the algorithm is to save energy and at the same time provide certain desired degree of coverage of the protected region at all times. However, all these algorithms analyze the degree of coverage from the perspective of target detection but not target tracking. One method of judging the effectiveness of a particular sensor deployment algorithm is by measuring the worst and best case coverage. [38], [39] provide algorithms to measure the worst and best case coverage based on Voronoi diagrams. In [40] the authors analyze worst case coverage (also called the breach path) in case of directional field-of-view sensor networks.

Line in the sand [41] system is a prototype model that can detect and classify up to three different target types. In [41], the authors discuss various issues in developing such systems, largely emphasizing data fusion algorithms. VigilNet [42] is a real time large-scale sensor network system that can track, detect and classify the targets in a timely and energy-efficient manner. In [42], the authors perform mathematical analysis of various delays and accuracy of the system. Both the above systems rely on mutual co-operation of group of clusters. They both assume the availability of a clustering algorithm. In [43], a target detection algorithm localizes a sound source using triangulation based on the acoustic measurements made by a group of three sensors.

Once again the existence of clustering algorithm is assumed. [44] evaluates three different architectures for fusing data collected by the sensors. The three schemes analyzed are a centralized scheme, a progressive scheme and a distributed scheme. A centralized source number estimation scheme is a processing structure in which all sensors send their raw data to a central processing unit where source number estimation is performed. A progressive source number estimation scheme is a processing structure that a group of sensors update the source number estimation result sequentially based on each sensors local observation and the partial estimation result from its previous sensors in the sequence. So, the information transmitted through the network is the estimation result or partial decision. Finally, a distributed or cluster based source number estimation scheme is a structure including two levels of processing: source number estimation within each cluster and decision fusion between different clusters. The authors conclude that the cluster-based distributed approach using the progressive intra-cluster estimation has the best performance in the sense that it can provide much higher detection probability than the centralized approaches, while at the same time occupying the least amount of network bandwidth and consuming the least amount of energy. [45] introduces Markov chain Monte Carlo data association algorithm to track an unknown number of targets. The algorithm once again relies on the existence of a clustering algorithm.

The clustering algorithm presented in this work has the ideal features pointed out in [44]. The algorithm is distributed in nature and allows for intra cluster data aggregation. The intra cluster data aggregation is made possible by the overlapping nature of the clusters. This also leads to redundancy and increases the success rate of target detection.

Chapter 4
Problem Statement

In this chapter we identify and analyze various issues that make the current target tracking system unreliable. The goal of the target tracking and detection system is to accurately and quickly predict the location and trajectory of a target so that it can be visualized by the camera. We also want the system to efficiently utilize the resources of the power constrained sensor nodes. The Target tracking application has two modules. The first module deals with successfully being able to detect the target and the second module deals with being able to transmit the target video reliably to an external agent.

For the prototype system, we have made a few assumptions to facilitate development and deployment. First, we assume a static network configuration of the nodes. The topology and node roles are statically defined. Clusters are composed beforehand using the IP addresses of the sensor nodes. Secondly, we assume time synchronization among the nodes in the system. Time must be synchronized so that CPA time of sensor nodes can be analyzed correctly. Comparisons are made based on a unified time to determine absolute time differences. Thirdly, we assume that general acoustic information about potential targets is available so that appropriate detection settings can be found. Suitable acoustic thresholds for the target and the environment need to be known in order to correctly trigger the detection mechanism. Fourthly, we assume only a single target is detected. The current system does not support multiple targets crossing the sensor field. The next two sections discuss in details some of the issues that make the system unreliable.

4.1 Basic Concepts of Target Detection Using CPA Algorithm

We estimate target location using closest point of approach (CPA) in distributed acoustic sensor networks. Sensor networks consist of a large number of inexpensive wireless sensor nodes. These sensor nodes can be distributed densely over the region of interest. Each sensor node in the sensor field generates a CPA data, which is a type of time-varying spatial signature of the moving target. When a target moves by one sensor node, the sensor node gets a series of time-varying signal using the average received energy. The signal reaches the peak when the target arrives at the nearest point to the sensor node. The sensor node then records the time t, which is relative to the peak. CPA data is composed of time t and the location of the sensor node. The location of each node can be determined using GPS.

4.2 Reliable Target Detection

One of the basic requirements for target tracking is reliable detection of targets by minimizing false positive and false negatives. Once this is achieved, then the CPA algorithm can be used to more accurately determine the target velocity and location. The two main causes of problems in target detection are the effect of background noise and wind sound on the acoustic sensors. For the acoustic sensor network, to successfully detect a target, the most important task is to distinguish the target sound from the background noise. One way to distinguish the target sound from the background noise is by setting a threshold level for each acoustic sensor. The CPA algorithm processes only those sound values that are above the threshold. But arriving at a threshold value is extremely difficult because of the ever changing nature of the back ground noise. The primary source of background noise is wind and stray events in the immediate neighborhood. Winds are omnipresent in the outside field, and the audio amplitudes of wind are always very high. In our real world experiments, we experimented with several mathematical models to arrive at a threshold value for the acoustic sensor so as to filter out background noise. Though these techniques help to reduce the effect of background noise, they do not completely eliminate errors generated by the back ground noise. The accuracy of the CPA algorithm and the effect of different errors on the accuracy of the overall system has been outlined in [46], [47] and [48]. These issues act as bottleneck at the cluster level. Further, in the real world to be able to track a large geographical area, we need to address issues related to organization of the sensors deployed in an effective way so as to save energy.

Finally, though the target might be traveling in a straight line for the perspective of any one cluster, it could be actually traveling in a curvilinear path. In [49] we evaluated the performance of single cluster algorithm. In a single cluster algorithm each node is allowed to form only one cluster. This limits the accuracy of the target detection system. Hence the clustering of sensors should be able to track such nonlinear paths. Furthermore the clustering algorithm should be able to reconfigure in the event of node failures.

We intend to improve the tracking capability of the over all system by making the best use of the sensors deployed. This implies the need for a way to measure the performance of the clustering algorithm. At first sight, it might appear that the total number of clusters formed would be best parameter to compare. But this is not true because some of the clusters may be redundant and may not provide any new information about the field to be tracked. We choose the following metrics to evaluate the performance of the clustering algorithm.

- Average Probability of Detection: This parameter estimates the average probability with which a target appearing at a randomly chosen point in the sensor field can be detected. A higher value implies a better clustering algorithm.

- Degree of Coverage: This parameter represents the region of the sensor field that is covered by at least one cluster. This parameter ensures that no region in the sensor field is left untracked. A higher value implies a better clustering algorithm.

- Breach Weight: Represents the minimum distance from sensors that an agent traveling on any path through the field A, from I to F, must encounter at least once. Lower breach weight represents better coverage.

- Support Weight: Represents the maximum distance from sensors that an agent traveling on any path through the field A, from I to F, must encounter at least once. Lower support weight represents better coverage.

- Protocol Overhead: Since sensors are energy constrained, we would like to see the cost of tracking a region. We would measure the amount of energy consumed to maintain the clusters. Also the energy needed to reorganize the clusters in the event of node failure is important.

Since each sensor has multiple neighbors, one important criteria to decide is coming up with heuristics a cluster head should adapt in choosing its member nodes. A highly irregular pentagon has the disadvantages that some regions enclosed by the pentagon are much better tracked than others. By symmetry a regular pentagon offers a more uniform tracking probability. we need a metric to measure the roughness or irregularity of a pentagon. One way to measure the irregularity of a polygon's perimeter is

$Irregularity = (MaxDim * BoundLen)/Area$

Where,

MaxDim represents the length of the longest diagonal from and to all polygon angles.

BoundLen represents the perimeter of the pentagon.

Area represents the area of the pentagon.

The other way to measure irregularity is by using the classical isometric inequality for an n-sided polygon P_n.

$$L_n^2 - 4d_nA_n \geq 0 \tag{4.1}$$

Where,

L_n is the perimeter of the perimeter of P_n

A_n is the area of the domain enclosed by P_n

$d_n = n \tan \frac{\pi}{n}$

Equality holds good for a regular polygon. In this work we choose Equation 4.1 to determine the degree of irregularity of a polygon.

4.3 Increasing Delivery Ratio in Wireless Multi-Hop Sensor Networks

Once the above reliability problems are addressed, the other components of the target detection and tracking system will be rendered more reliable. For instance, reliable target detection and reliable delivery of these detection and CPA sensor data to the cluster head will enable it to collaboratively compute the target location and velocity accurately. It will then reliably notify the camera sensor to track the target more accuracy. In [50] and [51] we evaluated the performance of directed diffusion and it can be concluded that directed diffusion though a scalable solution for wireless sensor networks does not address the real time and reliability issues of video sensor networks.

We conducted more field experiments to test the reliability of sensor data dissemination in wireless multi-hop sensor networks. In our experiments, we found that wireless link quality (using IEEE 802.11b Orinocco pc cards) may vary drastically based on a number of environmental variables, such as time of the day, surface materials of ground and buildings, distance of the device from the ground, power level of the device, distance between sender and receiver, etc. Sometimes packet loss can be high even when the distance between the sender and receiver is only 25 meters, whereas at other times, packet delivery ratio can be very high even when the distance if about 100 meters. Because of such erratic nature of reliability of wireless transmission and because collaborative target detection and tracking algorithms critically depend on reliable dissemination of sensor information, it is very important for wireless networking to be very reliable even when the reliability of each wireless link is low.

Intuitively the best way to improve reliability is

- Deploy denser sensor networks, or
- increasing the number of sensor nodes in a cluster.

Increasing the density of the sensor networks will ensure shorter distances between sensor nodes and higher packet deliver ratio. In the field experiment, instead of forming a cluster of four nodes for target detection, we will use six nodes, where the two redundant nodes are placed in the center to improve communication from the sensors to the cluster head. This will ensure that when a sensor node has CPA data to send, that directed diffusion will be able to forward packets more reliably through each link in the wireless network.

An alternative method is to place the sensor nodes closer to each other and reduce their communication distance to increase packet delivery ratio. This approach however has the disadvantages in that placing sensor nodes closer to each other will reduce the accuracy of the algorithm to compute target location and velocity.

The other method to increase communication reliability is to increase the number of sensor nodes in a cluster so that four sensor nodes report the CPA time to the cluster head instead of three. If all four report their CPA time, then the cluster head will drop the last one. If one of the sensor fail to send the CPA time through directed diffusion, then there are at least three more sensors that can report their CPA time. The cluster head will use the first three reported CPA data to calculate the target location and velocity. But this increases the cost of maintaining the cluster as we need to maintain a cluster of six sensors instead of 4 sensors. This would reduce the life time of the network.

For reliable multi-hop wireless communication, we use an automatic technique for recovery from communication failure by redundant neighboring nodes that overhears communication streams. By overhearing data stream flows along the main path, neighboring nodes may provide alternate path whenever wireless links along the main path fail. This method will enable reliable multi-hop wireless communication. Further, since this technique tries to provide reliability by having a packet recovery mechanism at every link, it is highly scalable.

Chapter 5

Architecture of Target Tracking System

We have developed a system for target detection and tracking using a collaborative sensor network. The architecture is built on the directed diffusion protocol which runs over the IEEE 802.11 MAC protocol. Sensors detect and analyze acoustic data to determine the closest point of approach (CPA) time of the target. Intuitively, this is the time when the target was loudest and therefore was closest to the sensor node. Sensors send CPA time information to the task group leader, the cluster head, for analysis. The cluster head uses the CPA algorithm described in [1] to predict the position, velocity, and direction of the target. This information is sent to a camera control application on an IP network though a node that bridges the diffusion and IP networks. The camera control node pans and zooms a camera to view the target's current location. Video and/or images of the target may then be captured for further analysis by a video capture application.

5.1 Architecture

The target detection and tracking system is composed of six components that are inter-networked together using IEEE 802.11, IEEE 802.3, IP, and directed diffusion. Each of the six components represents a role or a function which is performed by one or multiple computers. Figure 5.1 illustrates the architecture of the target detection and tracking system. Nodes serving as sensors record acoustic data in order to detect an event of interest and calculate the CPA time of that event. The cluster head is the leader of the target detection group. The node acting as cluster head receives CPA data from the sensors, executes the CPA target tracking algorithm, and sends the result to the bridge node which then forwards target location information from the diffusion network to the IP network. Specifically, the bridge sends data to the camera control PC which directly issues movement (pan and zoom) commands to the camera. Video or images are captured from the camera by the video capture PC. The sensing system is remotely controlled and monitored by the system control PC. This computer issues commands to nodes in the diffusion network.

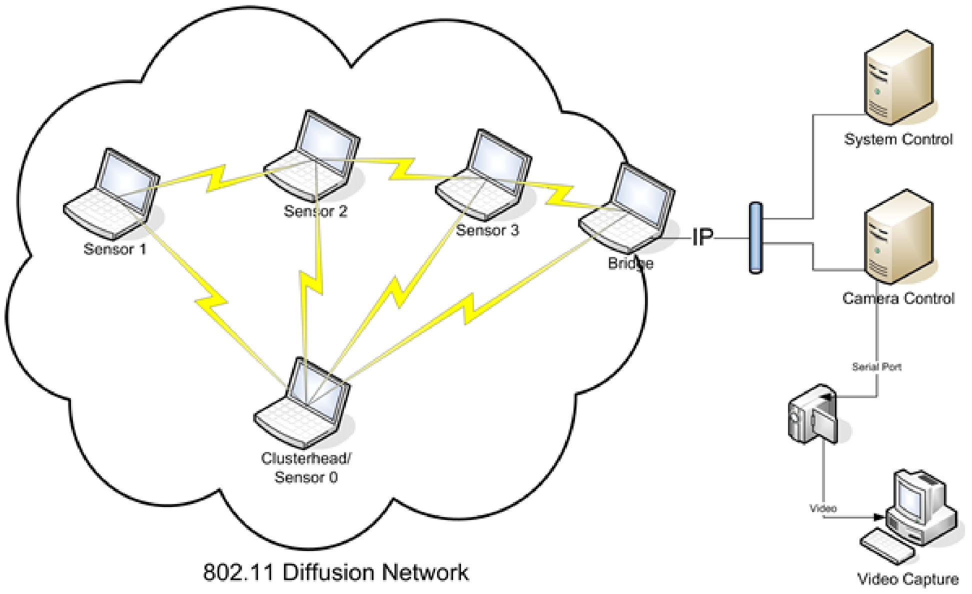

Figure 5.1: Target Tracking System Architecture.

5.1.1 Sensors

A sensor node monitors for targets, computes the CPA time, and reports the results to the cluster head. Initially, sensors are in the target monitoring mode where they continuously monitor for acoustic events. Once strength of the received acoustic signal exceeds a threshold, the sensor node switches to target detection mode. The node records sound for some short period of time (e.g. 3 seconds) and then analyze the sample to determine the CPA time of the target. The CPA time is the time when the maximum volume sound was recorded. After the CPA time has been calculated, the node sends its location, the CPA time, and the maximum acoustic signal strength of the event to the cluster head.

5.1.2 Cluster Head

The cluster head analyzes CPA time data from the sensors in order to determine the position, trajectory, and velocity of the target. Upon receiving the CPA time and location information from four sensors, the cluster head executes the CPA target tracking algorithm [1] to compute the location, speed, and direction of the target. This data, along with the CPA time of the cluster head, is transmitted to the bridge node.

5.1.3 Bridge

The bridge interconnects the diffusion network to the IP network. It forwards target location information from the cluster head to the camera control PC. The bridge represents the camera controller to the diffusion network. Its primary task is to receive packets destined for the camera and convert them from diffusion packets to IP packets. This involves subscribing to CAMERA data on the diffusion network and sending IP packets to the camera control node on the IP network. The bridge must extract the data from attribute vectors and repack it into an IP packet.

5.1.4 Camera Control

The camera control PC receives packets from the bridge and then pans and zooms the camera to point it at the location as contained in the packet. The camera controller resides on a wired Ethernet network. It listens for packets from the bridge on port 8899 and issues movement commands to the camera over the serial port based on the location and trajectory of the target. The camera pans and zooms according to the target data computed by the cluster head.

5.1.5 Video Capture

The video capture computer is responsible for interfacing with the video output of the camera. The output may be captured as a video or individual frames may be saved as images. Image analysis software could subsequently be used for more advanced target identification.

5.1.6 System Control

The system control PC manages the execution of the remote nodes. This includes starting, monitoring, and stopping processes. The system control computer uses SSH to remotely login and issue commands on the sensor, cluster, and bridge nodes. We have developed scripts to handle common tasks of the sensor system (e.g. starting and stopping diffusion). Since any command may be given over SSH, the system control PC has complete control of all the nodes in the diffusion network. The system control node may also be used to manage the camera control PC.

5.2 Implementation

The implementation of the sensor, cluster head and bridge applications is based on the implementation of directed diffusion by the Information Sciences Institute (ISI) [52]. The diffusion API developed by ISI provides convenient methods for developing diffusion based applications. The core functionality involves publishing and subscribing to named data. Data is structured as attribute-value pairs where an attribute is defined by a numeric key and the value is the data itself. Table 5.1 lists the key values of attributes used by the diffusion applications. These values must be unique, so they must be known for future development of the system. The last column contains the actual C++ data type that the diffusion attributes contain. To avoid loss of precision, data of type double and long were transmitted as BLOB_TYPE since they were too long to be stored in a predefined diffusion data type.

Table 5.1: Diffusion Attribute Names and Keys

Attribute	Key	Diffusion Data Type	C++ Data Type
CPA Time	6000	BLOB_TYPE	struct timeval
Time Stamp	6001	BLOB_TYPE	struct timeval
Task Name	6006	BLOB_TYPE	char []
Signal Strength	6007	BLOB_TYPE	int
Target Lat	6008	BLOB_TYPE	long
Target Long	6009	BLOB_TYPE	long
Target Speed	6010	BLOB_TYPE	double
Target Slope	60011	BLOB_TYPE	double

The API requires a configuration file, config.txt, to define the neighbors of the node. The config.txt file is in the following form:

[ip address] [port] [receptionRate]

[ip address] [port] [receptionRate]

where [ip address] is the IP address of a neighbor, [port] is the port on which diffusion is running on that neighbor and [receptionRate] is the probability that a packet is received. A value of 100 means that no loss is caused by diffusion. Each line of the file specifies a different neighbor. The default location of this file is the home directory.

5.2.1 Sensors

The sensor program monitors for a target, computes the CPA time, and sends it to the cluster head for analysis. Sensors parse a configuration file on startup which specifies the detection threshold and the monitoring duration. By default, this file is named sensor.txt and resides in the same directory as config.txt. The sensor file is in the following format where [threshold] is a value between 1 and 32000 and [time] is in seconds.

Threshold: [threshold]

RecordingTime: [time]

SSR: [ssr]

The value threshold value is the sound intensity at which the sensor changes from monitoring mode to detection mode. It should be chosen based on the acoustic properties of the target and environment. The recording time represents the time period over which the sensor records the target and computes a CPA time. The CPA time is the time when maximum sound value was recorded. SSR is a signal strength ratio which normalizes the sound intensity of each sensor to the clusterhead. This is necessary so that the clusterhead can make calculations using consistent sound intensities from each sensor. The SSR is calculated offline using a recorded sound.

The sensor must also know its own location. This is specified in the position.txt file which is located in the same directory as the other configuration files. The position file is written in the following format.

Latitude: [latitude]

Longitude: [longitude]

Once the CPA time has been determined, the sensor reports the information, along with its position, to the clusterhead. The sensor program sends data to the clusterhead according to the following publication definition (as specified in the directed diffusion protocol):

TaskNameAttr IS ”CPA”

CPATimeAttr IS [cpaTime]

SignalStrengthAttr IS [maxSignalStrength]

TimestampAttr IS [now]

The task name attribute identifies the type of data flow as CPA. The CPA time and maximum signal strength are used by the clusterhead for target localization and tracking. A timestamp is also transmitted in order to measure the latency of the network.

5.2.2 Clusterhead

The clusterhead receives data from the sensors, processes it, and sends the results to the bridge. The clusterhead program subscribes to data with the following subscription:

TaskNameAttr IS ”CPA”

Upon receiving CPA data from four nodes, the cluster head executes the CPA target tracking algorithm. The node working as the cluster head also serves as a sensor. Interprocess communication is handled by the diffusion routing core. Local messages are forwarded from the sensor process to the cluster head process on the cluster head node.

Messages are sent to the bridge using the publication definition shown below.

TaskNameAttr IS ”CAMERA”

TargetLatAttr IS [latitude]

TargerLongAttr IS [longitude]

TargetSpeedAttr IS [speed]

TargetSlopeAttr IS [slope]

CPATimeAttr IS [cpaTime]

TimeStampAttr IS [now]

The task name attribute identifies the type of data flow as CAMERA. The target latitude and longitude attributes are the predicted position of the target. The target's speed is measured in distance units per second. The distance unit is determined by what unit the sensor positions are measured. The slope of the target represents its direction of movement. Note that [cpaTime] corresponds to the CPA time of the sensor process running on the cluster head. It represents the time at which the target was located at the predicted position. As in the sensor, the timestamp attribute is used to measure the network delay.

5.2.3 Bridge

The bridge receives packets from the cluster head, converts them to IP packets, and forwards them to the camera control PC. The bridge subscribes to camera packets with the following subscription:

TaskNameAttr IS "CAMERA"

When camera packets are received, the bridge sends UDP packets to the IP address and port number specified in the bridge config.txt file. The configuration file is in the following format:

[IP Address] [Port]

The first line specifies the IP address of the camera control PC, and the port number identifies the port on which the camera control process is listening. The back end of the bridge is essentially just a UDP client application.

5.2.4 Camera Control

The camera control PC runs a simple UDP server listening for camera control packets. When packets are received from the bridge on the camera control port (8899 by default), the camera controller extracts the target location, velocity, and trajectory and moves the camera appropriately. The camera control application also receives the CPA time of the target. By calculating the delay between this time and the current time, the current position of the target can be projected. In order to move the camera correctly, the camera controller must know the position of the camera. This is specified in the cameraPosition.txt configuration file in the format shown below.

Latitude: [latitude]

Longitude: [longitude]

[Orientation Direction]

Orientation direction represents the direction in which the camera is pointing when in the home position. It must be one of the four cardinal directions (N,S,E, or W).

Camera movement is implemented using an open source library called EVILib which is written for Sony EVI video cameras. It is available at http:sourceforge.net/projects/evilib/. It offers a convenient API for accessing the camera over the serial port.

Camera movement occurs in two stages. First, the camera moves to view the current position of the target. We call this target acquisition. Secondly, the camera rotates to keep the target in view. We label this stage target tracking.

5.2.5 System Configuration

The target detection and tracking system uses two network protocols diffusion and IP. The network stacks are shown in Figure 5.2.

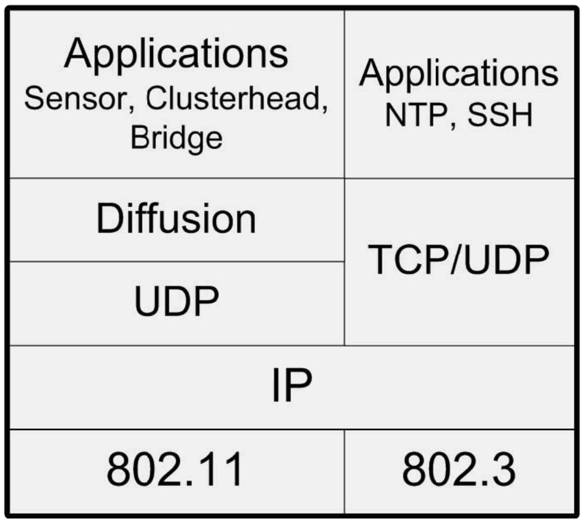

Figure 5.2: Protocol Stack.

IEEE 802.11 and 802.3 (Ethernet) underlie the communication links of the system. The sensors, clusterhead, and bridge communicate over 802.11. Wired Ethernet is used by the bridge, camera control, and system control machines. IP comprises the low level routing layer of the system and diffusion serves as a higher level network protocol. Diffusion and its applications run over UDP/IP. The non-diffusion applications use standard

Table 5.2: Parameter Settings

parameter	Value
Mode	Ad-Hoc
SSID	diffusion
Channel	11
Bandwidth	11Mbps

TCP/UDP as the transport layer. These include the system control (SSH) and camera control applications.

The diffusion nodes are configured to use the Ad-Hoc mode of 802.11. We used Lucent Technologies Orinoco 802.11b PCMCIA cards on the nodes. The bridge node had a built in 802.11 card which was used instead of an external card. The configuration of the wireless nodes is shown in Table 5.2.

Notice in Figure 5.2 that the diffusion runs on top of UDP and IP. This is because ISI diffusion is implemented with UDP as the link layer. As a result, diffusion executes over IP. While this may seem inefficient, it provides several benefits. First, by using diffusion over IP, wired Ethernet (802.3) can transparently be replaced by 802.11. The exact same version of diffusion can run over either MAC layer. This significantly eases configuration. Secondly, by running IP among the sensor nodes, time synchronization can be performed using standard NTP. Furthermore, standard SSH (over IP) can be used to remotely login and manage the nodes on the diffusion network. The use of IP as an underlying routing protocol significantly eases configuration and management of the system.

Although the network runs IP, the addition of diffusion, allows multi hop routes to be established dynamically. Diffusion establishes multi point to multi point data rows over multiple hops. This functionality would be even more advantageous in larger networks, i.e., networks with more nodes where a multiple hop route may be the only path between two nodes. Diffusion will deliver packets from the cluster head to the bridge even if the bridge is not directly connected to the cluster.

Chapter 6

Algorithm Design for Reliable Detection and Communication

In this chapter we describe in detail the two major contributions that bring the target tracking system much closer to reality. The first is a distributed clustering algorithm that organizes the deployed sensors into clusters of five sensor nodes. The algorithm is robust to node failures and in the event of a node failure it re-organizes the remaining nodes into clusters of five nodes by restricting the changes to a minimum.

The second contribution deals with improving the end-to-end reliability of Directed Diffusion so that the video recorded by the sensor can be delivered to an external agent with minimum effort in terms of energy. This is essential since sensors are low powered devices and the lifetime of the network is an important parameter for the applicability of the overall system.

6.1 Clustering Algorithm

In this section we introduce a clustering algorithm that groups deployed nodes into groups of five. The clustering algorithm can be made to operate in two modes: overlapping and nonoverlapping mode. The percentage of tracking region that two clusters share can be controlled by allowing adjacent clusters to have some nodes in common. Figure 6.1 shows the scenario where we have nonoverlapping clusters. In this case we see that we have only one cluster tracking a region. This implies in the event of failure we do not have a backup cluster to track the target. Further in the nonoverlapping mode of operation we can form only 4 clusters from the 15 sensors deployed. Whereas in Figure 6.2 it can be seen that we are able to identify 7 clusters for the same set of 15 sensors. The overlapping nature of clusters provides for redundancy and hence better reliability. Further, it enhances the ability of the system to track curvilinear trajectories.

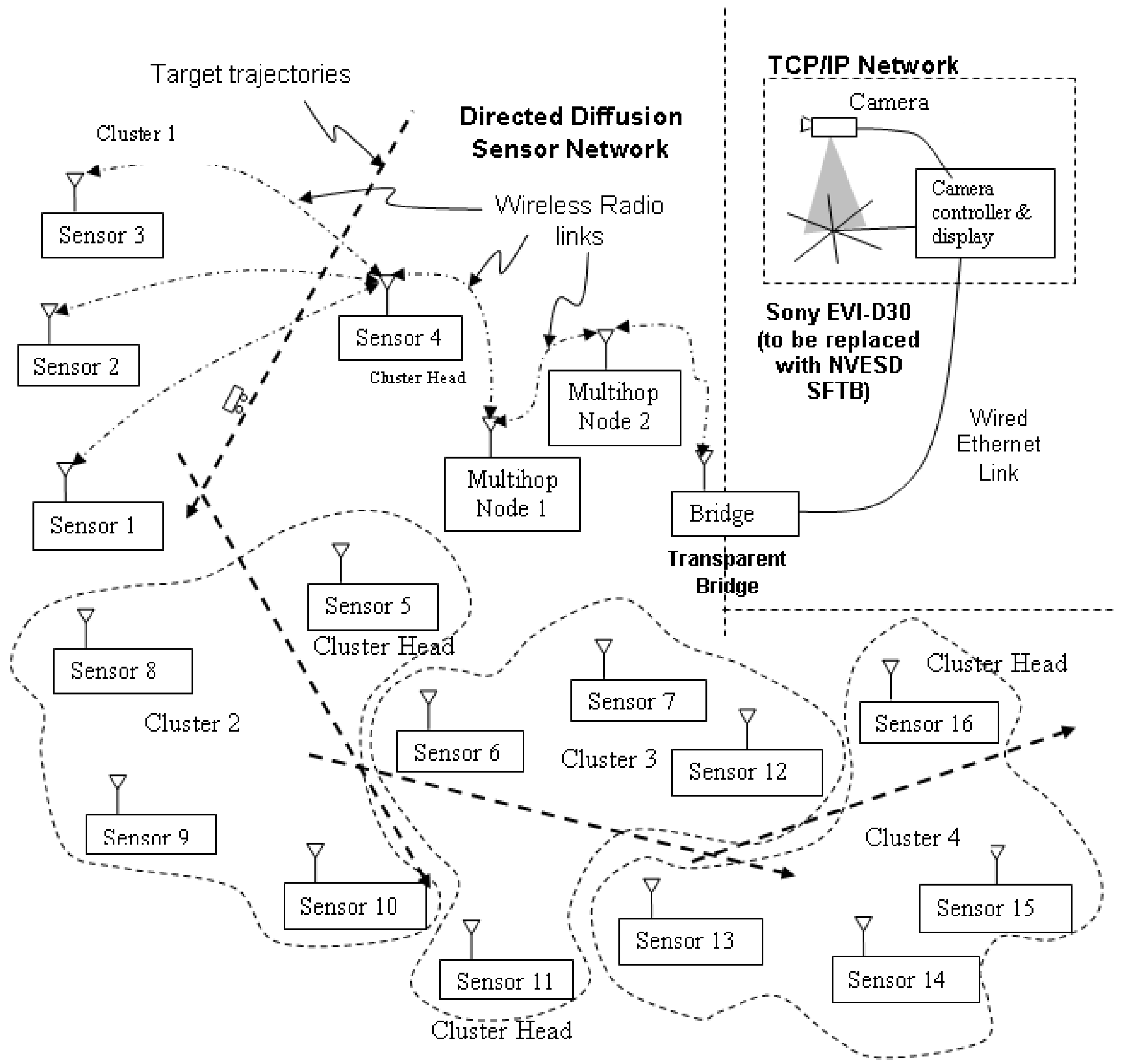

Figure 6.1: Non-Overlapping Clusters for Analyzing Non-Linear Trajectory of Moving Targets

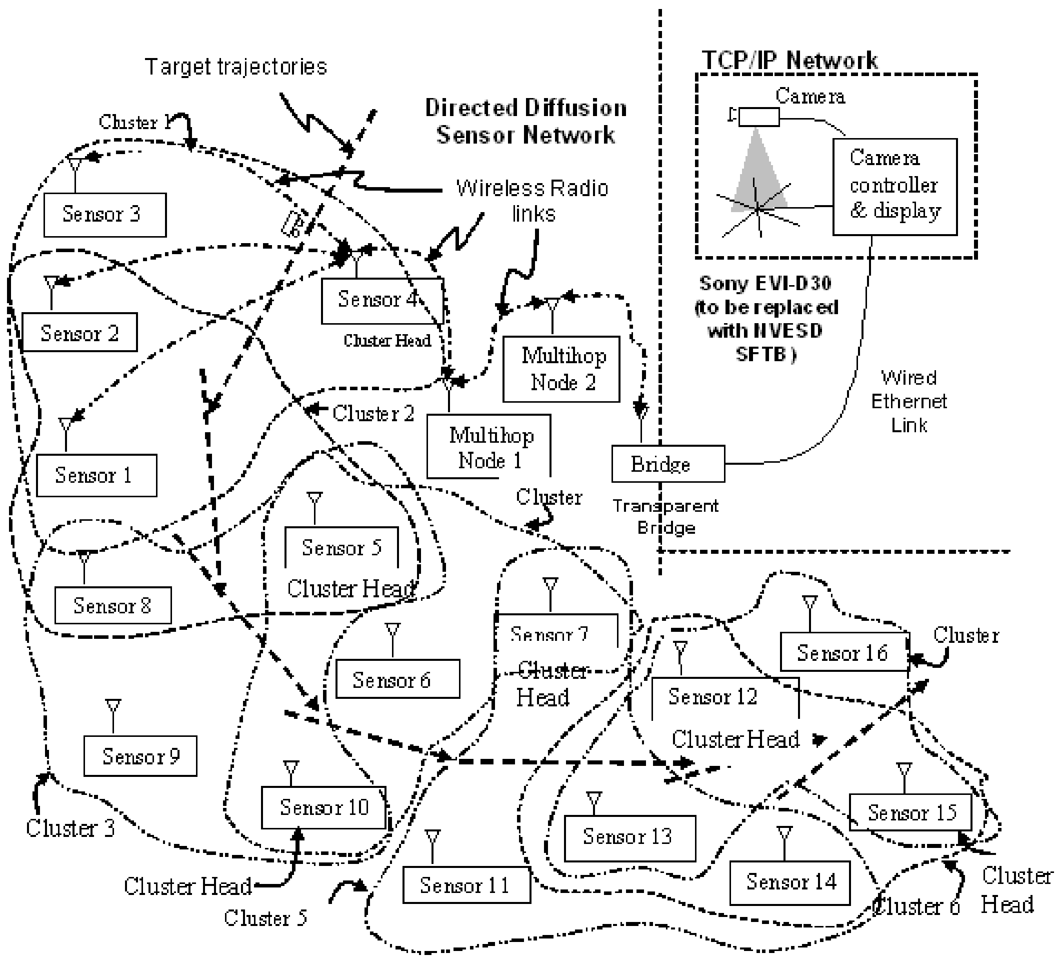

Figure 6.2: Overlapping Clusters for Analyzing Non-Linear Trajectory of Moving Targets showing more accurate and finer-grain non-linear trajectory of the target

6.2 Need for Redundant Clusters

In Section 2.3.1 we have seen that a single cluster tracking a region is reliable only to a certain extent. This is because the individual sensors that make up the cluster are prone to detection errors and we represent this probability of failure by *p*. Now we derive an expression for the probability of failure for a cluster to track an event can be obtained by evaluating the following two cases.

Case 1: Assume that the target takes path P1, i.e. 1 sensor on one side of the target trajectory and all the other sensors on the other side. Let *P1* represent the probability of failure.

$$P1 = 1 - probability\ of\ success = 1 - ({}_4C_3 * (1-p)^4 * p + (1-p)^5) \tag{6.1}$$

Case 2: Assume that the target takes path P2, i.e. 2 sensors on one side of the trajectory and the rest on the other side. Let *P2* represent the probability of failure.

$$P2 = 1 - probability\ of\ success = 1 - ({}_2C_1 * (1-p)^4 * p + (1-p)^5) \tag{6.2}$$

Since there are only 5 possible scenarios in which Case 1 can happen and 5 possible scenarios in Case 2 can happen, the probability with which a cluster can fail to detect an event is 0.5**P1* + 0.5**P2*.

Figure 6.3 represents the individual failure rates of each case and the overall failure rate with changing failure rates of individual sensors. A sensor fails to detect because of various factors, such as ambient noise and issues associated with the fine tuning of thresholds for the acoustic detector. The total success rate can be improved by having more than one cluster

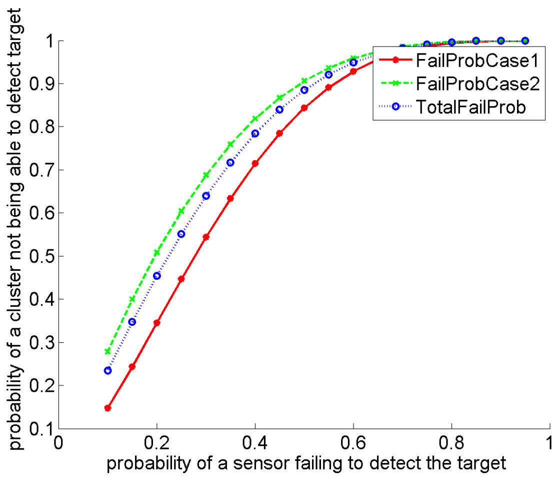

Figure 6.3: Failure probability of a cluster in detecting the target given individual sensor failure probability.

monitor a certain region. This makes it necessary to have overlapping clusters. If we have six sensors, then we can have six clusters such that any two clusters differ by at least one sensor. Incorporating new nodes and forming new clusters with different set of sensors can give a different perspective to evaluate the tracking parameters. Figure 6.4 shows how the total failure to track a target decreases as more clusters track the target. N represents the number of clusters that track a region. It can be seen that the failure probability can be reduced greatly by increasing the number of clusters tracking a target from 1 to 5.

In the algorithm presented in the next section, the amount of redundancy can be controlled by controlling the number of sensors two clusters can have in common.

6.3 Design Details of Clustering Algorithm

The clustering algorithm forms as many polygons as possible and prevents polygons with exactly the same set of sensors from forming. Ties are broken by giving preference to

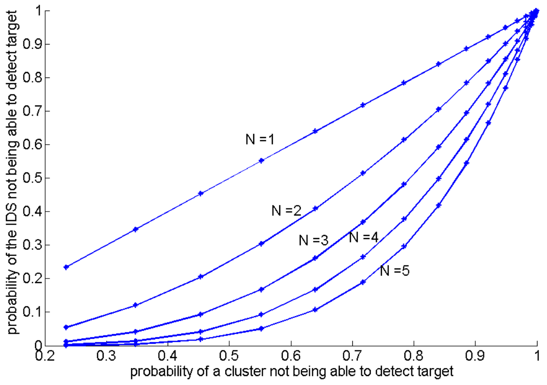

Figure 6.4: Improvement in the target detection as more number of clusters monitors a region.

the sensor with lower ID. Once formed, a cluster head remains in cluster head state until it detects that one of its member nodes is not responding. The cluster head node and the member nodes poll one other with HELLO messages to detect node failure. In the event of node failure, the cluster head disbands the cluster and moves to the initial state and starts all over again. Each sensor (cluster head) manages only one cluster at a time, but can be a member node to any number of clusters. The overlapping nature of the clusters improves the reliability of the system as the same portion of the field is monitored by more than one cluster. An overview of the protocol is shown in Figure 6.5.

The protocol has eight states and makes use of seven messages. The states are START, CHEAD, HELLO, POLY_REQ_SENT, POLY_INFORM_SENT, and CHEAD_LOST. A brief description of each state is given below.

- START: This is the initial state of the node soon after it has been deployed out on the field.

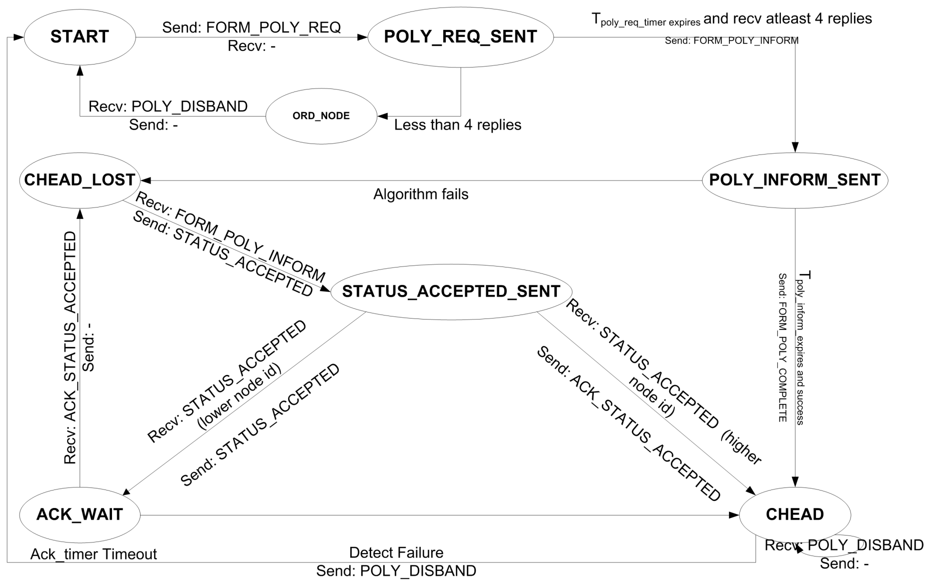

Figure 6.5: State transition diagram of the clustering algorithm.

- CHEAD: A node moves to this state on successfully forming a polygon. It receives CPA information from the member nodes and runs the CPA algorithm to determine the target parameters. It periodically sends out HELLO messages to verify that its member nodes are alive. A node in CHEAD state remains so until one of its member nodes dies.

- POLY_REQ_SENT: A node interested in forming a cluster broadcasts a FORM_POLY_REQ message and move to this state while it waits for replies from nodes that interested in forming a cluster.

- POLY_INFORM_SENT: Once a node decides on its member nodes, it broadcasts a FORM_POLY_INFORM message. This message carries the ID of all the member nodes.

- CHEAD_LOST: when a node that intends to form a polygon realizes that a similar polygon is being formed by another node (with a smaller ID), it gives up its attempt to form a polygon and moves to CHEAD_LOST state.

The seven messages are FORM_POLY_REQ, FORM_POLY_REPLY, FORM_POLY_INFORM, FORM_POLY_COMPLETE, STATUS_ACCEPTED, ACK_STATUS_ACCEPTED, HELLO and POLY_DISBAND.

- FORM_POLY_REQ: this is a request message broadcast by a node that intends to form a cluster soliciting replies from nodes that are interested in joining a cluster.
- FORM_POLY_REPLY: Upon receiving a FORM_POLY_REQ message, a node interested in joining a cluster replies with a FORM_POLY_REPLY message.
- FORM_POLY_INFORM: A node that has decided on the member nodes of its cluster broadcasts a FORM_POLY_INFORM. This message contains the IDs of the member nodes. The purpose of this message is to avoid duplicate clusters. Duplicate clusters are those clusters that have identical nodes.
- FORM_POLY_COMPLETE: This is a broadcast message sent by a node to inform its member nodes about the successful formation of the cluster. The sender of the message becomes the cluster head. The member nodes send raw data about any target they detect to the cluster head.
- HELLO: The cluster head and the member nodes make sure that the cluster is intact by periodically exchanging HELLO messages between them.

- POLY_DISBAND: Once a cluster head concludes that one of its member nodes is not responding, it disbands its cluster by broadcasting a POLY_DISBAND message. It then moves to START state and starts all over again.

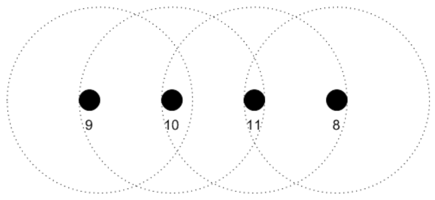

Figure 6.6: A special case of the clustering algorithm.

In the rest of the section, we explain the purpose of STATUS_ACCEPTED and ACK_STATUS_ACCEPTED message. The main purpose of the message is to reconfigure the clusters in the event of node failures and at the same time ensure that neither too many redundant clusters are formed nor too few clusters are formed. Too few clusters might lead to void region. Void region is the region that is not monitored by any cluster.

- STATUS_ACCEPTED: In Figure 6.6, assume that nodes 9, 10 and 11 all try to form clusters at the same time. Also assume that the set of member nodes of 9 and 10 differ by a single node and the set of members of 10 and 11 also differ by a single node. However, assume that the set of member nodes of clusters being formed by nodes 9 and 11 differ by 2 nodes. By virtue of lower ID, node 9 gets to form a cluster while 10 moves to CHEAD_LOST state. Assume that each of the nodes have exchanged the FORM_POLY_INFORM messages. By the nature of the node positions, node 9 is aware of the cluster being formed by 10; 10 is aware of the clusters being formed by both 10 and 11; and 11 is aware of the cluster being formed by 10. Now there is a tie between nodes 9, 10 and 11. Since the algorithm allows for nodes with lower ID to form clusters, only node 9 would be able to form a polygon. This because 10 would loose the race to 9 and 11 would loose the race to 10. We could better monitor the field if 9 and 11 can form clusters as their respective set of member nodes differ by 2 nodes. Hence, in such situations, 10 allows 11 to form a cluster by sending a STATUS_ACCEPTED message.

Before the STATUS_ACCEPTED message reaches node 11, assume that node 11 looses to node 8 (since 8 has lower ID, it has precedence over 11 in case of, tie.). In this case, we would end up with just two clusters formed by nodes 9 and 8. This might lead to a region between nodes 9 and 8 not being monitored. In order to avoid such situations, node 11 sends out a STATUS_ACCEPTED message to node 10, and node 10 goes ahead to form a cluster. Node 11 will then move to STATUS_ACCEPTED_SENT state.

- ACK_STATUS_ACCEPTED: While in STATUS_ACCEPTED_SENT state, if a node receives a STATUS_ACCEPTED message from a node with a lower ID, it replies with a STATUS_ACCEPTED message and waits for (ack_timer period) the lower ID to acknowledge the message with ACK_STATUS_ACCEPTED message. On the other hand, if it receives an STATUS_ACCEPTED message from a higher node ID, it moves to CHEAD state and the lower ID acknowledges with a ACK_STATUS_ACCEPTED message. This final step of the algorithm assures that nodes with a lower ID gets to form a polygon even in the presence of wireless losses, and, at the same time, assures that no void regions (regions that are not monitored by any cluster) are formed.

6.3.1 Metrics for Identifying the Member Nodes of a Cluster

From Figure 6.5 it can been that a node that intends to become a cluster head needs to have a policy to identify the best four nodes as its member nodes. The policy to identify the member nodes depends on the aim of the application. There are several parameters that can be optimized. Some of them are high probability of detection, low energy and cost, etc. The current implementation is flexible and allows for optimizing any of the parameters. In this work we aim to achieve 100% coverage with minimum number of clusters and the best possible detection probability. In order to achieve these goals we choose member nodes such that the resulting pentagon is highly regular. We measure the degree of regularity of a pentagon by using Equation 4.1.

6.4 Reliable Directed Diffusion Protocol

Directed diffusion uses delay as the metric to identify the best route between source and sink nodes. RDD proposes a data centric approach for route repair which makes use of backup nodes at each hop in order to reroute data packets in the event of link failure. Hence the end-to-end delay is not greatly affected.

On receiving either a positive reinforcement or data packet, a node identifies itself to be a *main* node for the corresponding message attributes. A main node is the node that forwards data packets for a certain attributes. We refer to the nodes that backup the main nodes as *backup* nodes. Since the role played by a node is based on the message attributes, a node can be a main node for one set of attributes and at the same time be a backup node for a different set of attributes. Every node maintains a list of all the attributes for which it is the main node and for those it is a backup node. These attribute value pair help in rerouting in the event of node and link failures.

We have defined two different modes of RDD: hierarchical and non-hierarchical. In hierarchical mode, each hop of the data path (identified by the diffusion protocol) is backed up by a set of backup nodes which, in turn, are backed up by another set of nodes. The level of hierarchy is chosen at the time of configuration. The default mode, non-hierarchical, is similar to hierarchical mode of level 1. In this mode, the backup nodes do not have any backup nodes. If the backup nodes fail to deliver a reroute data packet they simply broadcast it. Since sensor networks are dense there is good chance for a downstream backup node to receive the data packet. The downstream backup node will then make an attempt to deliver the data packet to a node along the original data path based on its attribute value pair.

6.4.1 Identifying Backup Nodes

The first step in the protocol is identifying backup nodes. This is done by modifying the basic diffusion protocol. The sources in the network reply to the interest packets sent by the sink nodes with exploratory packets. The sink nodes then reinforce the node from which it first receives the exploratory data. After a node transmits a positive reinforcement message it broadcasts a *backup positive reinforcement request* message to its 1-hop neighbors. All the nodes which receive the backup request message create a link in their routing table. This is illustrated in Figure 6.7 where node 10 is the source and node 19 is the sink while nodes 12 and 13 are the intermediate nodes. Node 13 forwards the positive reinforcement message that it receives from node 19 to node 12. Immediately after forwarding the positive reinforcement message it broadcasts a backup positive reinforcement request message asking nodes to backup the link 13 → 12. When Nodes 22 and 2 receive this message, they create a link entry in their routing tables as shown in Table 6.1.

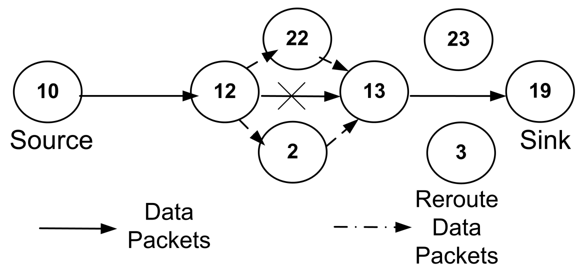

Figure 6.7: Routing in the event of Link failure

Table 6.1: Link entries at nodes 2 and 22

Link Type	Link ($S \rightarrow D$)	Attribute/Value Pair
POS_REINFOR	13 → 12	*AttrValue*

This happens at each hop along the positive reinforced path. But since Directed Diffusion sends out a positive reinforcement packet only once every 60 seconds this message does not lead to a significant overhead. When the source node receives the positive reinforcement message, it sends out data packets which retrace the reinforced path back to the sink. At each hop, the node that forwards the data packet, broadcasts a *backup data request* message. We reduce the overhead of the backup data packet by broadcasting it only once every K data messages. K is a configuration parameter and is used to control the message overhead. The nodes that receive the backup data request message create a link entry in their routing table. In Figure 6.7, node 12 broadcasts a backup data request message asking the nodes to backup link 12 → 13. Nodes 2 and 22 will then have two link entries as shown in Table 6.2, one corresponding to positive reinforcement from 13 → 12 and the other corresponding to data from 12 → 13. Nodes 2 and 22 conclude that they can backup node 13 for node 12 and send out a *backup data reply* message to node 12. Node 12 adds the address of all the nodes that willing to serve as backup nodes for node 13.

Table 6.2: Link entries at nodes 22 and 2

Link Type	Link ($S \rightarrow D$)	Attribute/Value Pair
DATA	$12 \rightarrow 13$	*AttrValue*
POS_REINFOR	$13 \rightarrow 12$	*AttrValue*

This implies that if node 12 fails to transmit a data packet to node 13 it can either transmit the data packet to nodes 2 or 22 who in turn would try to deliver the packet to node 13. In this way, every hop from the source to the sink will have identified some backup nodes by the time the first data packet is delivered.

6.4.2 Alternative Technique to Identify Back up nodes.

One way to reduce the message overhead due to back up positive reinforcement messages and backup data messages is by operating each sensor node in promiscuous mode. Operating every sensor node in promiscuous mode would mean every node is identical to broadcasting every message through out the network. This is because the MAC on receiving a RTS/CTS message, compares the destination address with its own address and determines whether the subsequent communication is intended for it or not. Hence when operating in promiscuous mode it would turn off this filtering mechanism and send every packet it receives to higher layers. This is similar to broadcasting every data and positive reinforcement message. Though a positive reinforcement may not cause significant protocol overhead (since it is transmitted once every 60 seconds), data message would lead to protocol overhead and reduce the available bandwidth of the sensor network. The cost of operating a node in promiscuous mode has been evaluated in detail in [53].

6.4.3 Failure Detection

For most network analysis, the packet loss rate is assumed to be constant and equal for every link. Based on this assumption, we can conclude that the path quality degrades exponentially as the path length (expressed as number of hops) increases.

However, in reality, the constant loss rate assumption is not true. From Figure 2.4 it can be seen that the link quality changes dynamically and degrades significantly with distance. Each link has a different packet loss rate that changes with respect to time. This implies that under certain conditions adding an extra hop between nodes 12 and 13 might actually improve the packet delivery ratio. The MAC attempts to retransmit an unacknowledged data packet until the number of retries reaches the maximum retry limit(defined by the specification). On reaching the maximum retry limit it informs the higher layer that the send operation failed and discards the packet. To handle failures in RDD, the MAC layer informs the network layer whether or not a packet was successfully delivered. The cross layer communication is in the form of MAC layer returning the sequence number of the packet that it failed to deliver. This mechanism is explained in detail in [54], [55] and [56]. Once the sequence number is returned, the network layer can retrieve the data packet from its cache. Since the Interframe spacing(IFS) is of the order of a few microseconds, the turnaround time involved in identifying a packet loss is extremely small and the size of the cache required is also extremely small. Hence, in our simulations, we do not consider the effects of the cache. This design makes it unnecessary for the receiver to have a NACK mechanism at the MAC layer and eliminates the need for the sender to maintain a data cache at the network level for longer intervals.

6.4.4 Link Failures

Assume that node 12 fails to transmit a data packet to node 13. The MAC layer informs the network layer about the failure. The network layer then looks for the possible backup nodes for node 13 and selects one of the possibilities (e.g. node 2). Node 12 sends out a *reroute data* packet to node 2. If the new link fails the next backup node will be used (e.g. node 22). This process is repeated until node 12 has exhausted all its backup options. Otherwise, node 22 then forwards the data packet to node 13. As soon as a main node receives the rerouted data packet, it converts it to a regular data packet and applies the usual forwarding rules. In Figure 6.7, when the link $12 \rightarrow 13$ fails, data packets would be rerouted either through 2 or 22.

6.4.5 Node Failures and Recovery (Non-Hierarchical)

RDD handles node failures in two ways: hierarchical and non hierarchical mode. Each main node keeps track of the number of consecutive attempts it has failed to deliver a packet to the next hop. If this count reaches a threshold (we set it to be three) the sender then assumes that the next hop is dead and permanently reroutes data packets to a backup node. In the example configuration, if node 12 fails to deliver three consecutive data packets to node 13, node 12 concludes that node 13 is dead. Node 12 then transmits a reroute data packet to node 22 which simply broadcasts the failed reroute data packet. If the density of the sensor network is high, backup nodes of the next link could receive the broadcast packet. In Figure 6.7, nodes 23 and 3 are the backup nodes for the next hop $13 \rightarrow 19$. On receiving the broadcast data packet, assume node 23 realizes that the sender failed to deliver it to node 13. Since, it is a backup node for the link $13 \rightarrow 19$, it transmits to node

19. The procedure continues until the packet is delivered to a main node or else no more backup nodes can be found. Node 12 requests the main node that would receive the next (fourth) reroute data packet to send out a positive reinforcement so that the route can be fixed locally. This is done by setting a flag in the reroute data packet. Once the route is fixed, the positive reinforcement packet and the subsequent data packets along the repaired path will identify the backup nodes for each hop as explained previously. In Figure 6.8, the dashed circle represents the transmission range of a node of identical color. Further in certain scenarios like the scenario shown in the lower half of Figure 6.8. In this case Nodes 2 and 3 are not in hearing range of one another. The only way to reroute the data packet is through 4.

6.4.6 Node Failures and Recovery (Hierarchical)

Broadcasting data packets is expensive since all the nodes in the communication range have to process it. In hierarchical backup mode, the backup nodes recursively establish backup nodes. There are situations where a multi-tier backup is necessary to route packets around a node failure. For example in the lower half of Figure 6.8, $10 \rightarrow 12 \rightarrow 13 \rightarrow 19$ is the path established by diffusion between the source node 10 and the sink node 19. Nodes 2 and 3 are backup nodes for the links $12 \rightarrow 13$ and $13 \rightarrow 19$ respectively, i.e., if node 12 fails to transmit a data packet to 13 it sends out a reroute data packet to 2. Assume that node 13 is dead. In this case, node 2 would not be able to transmit it to node 13. Hence, when node 2 fails to transmit the reroute data packet, in the absence of hierarchical backup, node 2 simply broadcasts the reroute data packet hoping that some downstream backup node receives it. In the above scenario, there is no downstream backup node within the transmission range of node 2, and thus, the route repair algorithm fails. Hierarchical backup mode overcomes this limitation.

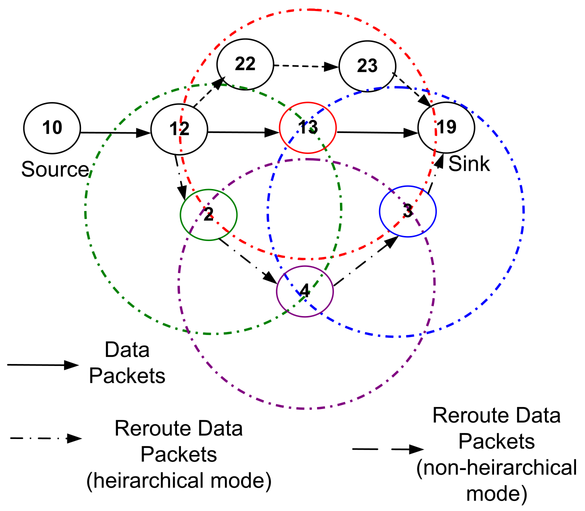
22
23
10
12
13
19
Source
Sink
2
3
4
Data Packets
Reroute Data Packets (heirarchical mode)
Reroute Data Packets (non-heirarchical mode)

Figure 6.8: Rerouting in the event of node failures

In hierarchical backup mode, on accepting to be a backup node for link 12 $\rightarrow$ 13, node 2 broadcasts a backup data requesting for backup of link 12 $\rightarrow$ 13. Similarly node 3 broadcasts a backup data requesting for backup of link 13 $\rightarrow$ 19. Node 4, which is within the transmission range of node 2 and node 3, adds the backup data request into its link cache. It further concludes that it can act as backup for the link 2 $\rightarrow$ 3 and sends out a backup data reply to node 2 to confirm that it can be used as a backup for node 13. On receiving the reply, node 2 adds node 4 as a backup node for node 13. If node 2 fails to transmit reroute data packet to node 13, it forwards the reroute data packet to node 4 which in turn forwards it to node 3 which can now forward it to node 19. Although hierarchical mode requires more setup overhead, it provides much more robust recovery capabilities.

An alternative technique is to use localized flooding. However, for efficient flooding it is necessary to determine the correct hop count for any given random topology. This is because in sensor networks a node does not have any topological information of the network.

6.5 Design Details of Reliable Directed Diffusion

Reliable directed diffusion adds four new messages to the existing directed diffusion protocol, and each of these message types adds four fields to the existing message structure. The new fields are *Source*, *Destination*, *FailedNode*, and *NodeType*. The *Source* and *Destination* fields contain the source and destination link layer addresses of the two nodes on the current data flow. The *FailedNode* field refers to the address of a downstream node that a main node has identified to have failed. The *NodeType* field holds the status of the message source, either MAIN_NODE or BACKUP_NODE. Every node besides the main nodes also maintains two link caches. One cache stores the list of links for which it has received a backup request while the second cache stores the list of links for which it is a backup node. The new messages are:

- BACKUP_POPSREINF: This message is broadcast by a node soon after it sends out a POSITIVE REINF message. The *Source* and *Destination* fields refer to the source and destination of the preceding POSITIVE REINF message. In this case, *NodeType* is set to MAIN_NODE.

- BACKUP_DATA: This message is broadcast by a node soon after it sends out a DATA packet. The *Source* and *Destination* fields refer to the source and destination of the preceding DATA message. The *NodeType* field is set to MAIN_NODE or BACKUP_NODE depending on the status of the requesting node.

- BACKUP_DATA_REPLY: This is a unicast message sent from a backup node to a main node informing the main node that it can be used to reroute DATA packets.

- REROUTE_DATA: This is a unicast message that a main node sends to one of its backup nodes if its transmission to a downstream node fails. The *Source* and *Destination* fields at all times carry the source and destination addresses of the link along which DATA packets are expected to travel.

6.6 Protocol Overhead of Reliable Directed Diffusion

Reliable directed diffusion introduces four new messages. Three of them BACKUP_POS_REINF, BACKUP_DATA and BACKUP_ DATA_REPLY are necessary to maintain alternate routes at each hop and add to the protocol overhead. In this section we evaluate the protocol overhead with increase in the path length.

Let n be the path length measured as number of hops. The protocol tries to identify backup nodes at every hop. Hence, backup positive reinforcement and backup data messages are generated at every hop. A node broadcasts a backup positive reinforcement message every time it needs to send a positive reinforcement message. Assume that the application lasts for T seconds, and diffusion send out a positive reinforcement every $T_{ExploreDataDelay}$ seconds. The total number of backup positive reinforcement messages broadcast is $n \cdot (T/T_{ExploreDataDelay})$.

Similarly, a node broadcasts a backup data message every time it has a data packet to send. However, to save energy, nodes generate backup data messages only for k percent of the actual data messages. If R is the data send rate of the application and the application lasts for T seconds, then the total number of backup data messages is $k \cdot R \cdot T \cdot n$.

Let N be the average number of nodes that volunteer to backup at each hop. On receiving a backup data request, nodes that volunteer to serve as backup nodes reply with a backup data reply message. If N nodes volunteer to backup, the total number of backup data reply messages generated is $N \cdot (k \cdot R \cdot T \cdot n)$ since $k \cdot R \cdot T \cdot n$ is the total number of backup data messages. Thus, the total number of additional messages generated by the Reliable Directed Diffusion is given by Equation 6.3

$$\begin{aligned} n \left\lceil T/T_{ExploreDataDelay} \right\rceil + kRTn + N(kRTn) &= \\ n \left(\left\lceil T/T_{ExploreDataDelay} \right\rceil + kRT + NkRT \right) &= \\ n \left(\left\lceil T/T_{ExploreDataDelay} \right\rceil + kRT\,(N+1) \right) & \end{aligned} \tag{6.3}$$

The parameters T and R are application dependent and k is a configurable parameter. N depends on the node density and is defined by sensor deployment. Thus T, R, k and N remain fixed for a given experimental setup. From Equation 6.3 it can be seen that protocol overhead increases linearly with respect to the path length (n).

In Figure 6.9 we compare the protocol overhead of Reliable Directed Diffusion and Directed Diffusion for N = 3. We evaluate the two protocols for different path lengths n.

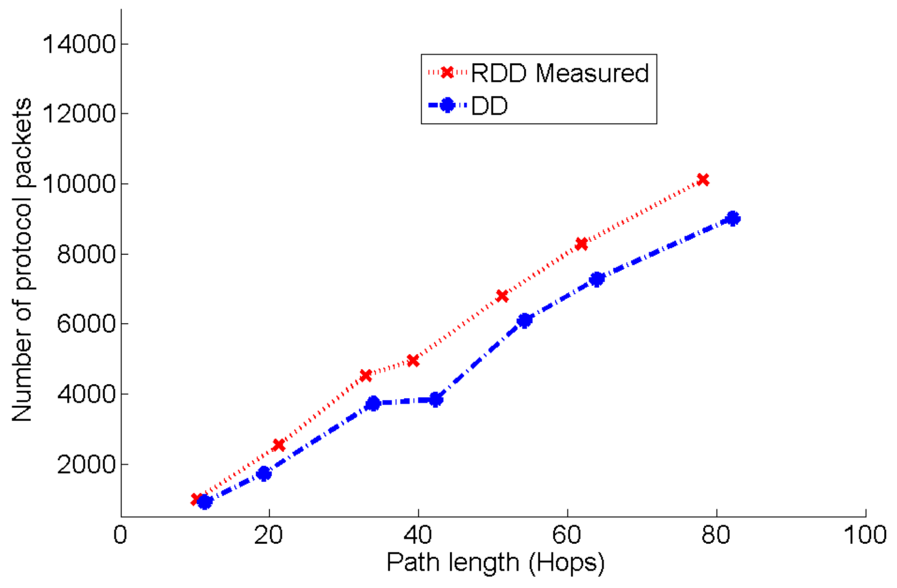

Figure 6.9: Protocol Overhead of RDD and DD.

Chapter 7

Implementation

7.1 Robust Clustering Algorithm

We implemented the clustering algorithm in Ns2 [57]. This required the creation of a new agent called Clustering Agent. In this section we explain in detail the implementation of the Clustering Agent. The Clustering Agent implements Clustering Algorithm by making use of five timers and nine helper functions.

7.1.1 Timers

In this section we explain the significance of various timers used to implement the Clustering Algorithm. We also explain the data structures used in the process of implementing the Clustering Algorithm.

Polygon Request Timer

A sensor that tries to become a cluster head broadcasts a FORM_POLY_REQ message to its neighbors to solicit for members to form new clusters. Soon after sending out the FORM_POLY_REQ message its starts the Polygon Request Timer. The nodes that are interested in joining a new cluster reply with a FORM_POLY_REPLY message. The node that intends to form a new cluster records the replies in its neighbor table. The neighbor table is a linked list. Each element of the list is defined as:

```
class Vertex {
    int nodeID;
    double nodeLocX;
    double nodeLocY;
};
```

- `nodeID` - The ID of the node that sent the reply message
- `nodeLocX` - The X co-ordinate of the node that sent the reply message
- `nodeLocY` - The Y co-ordinate of the node that sent the reply message

Polygon Inform Timer

A node that had received replies to the FORM_POLY_REQ message, evaluates different possibilities of forming a cluster and using heuristics it picks four member nodes. The node then broadcasts a POLY_INFORM_MESSAGE. Soon after broadcasting the message it starts the `Polygon Inform Timer`. On receiving the message nodes reply giving information about the clusters that they have heard of. So during this period the sending node gets to know if a similar cluster is being formed in its vicinity. The information gathered about the clusters in a node's vicinity is stored in a linked list. Each element of the list is defined as

```
class Polygon {
    int CH;
    int members[POLYGON];
    double ts_;
};
```

- `CH` - The ID of the Cluster Head
- `members[POLYGON]` - This is an array storing the node id of the member nodes of a cluster

- `ts_` - Time stamp indicating when the message was received

Ack Timer

In case two nodes attempt to form the same cluster, the node with lower id is given preference. But in some situations the node with lower id might loose its claim to be cluster head to a node with even lower id. In such cases the node with lower id sends out a unicast message POLY_STATUS_ACCEPTED message to the higher node id authorizing the higher node id to form cluster. Soon after sending the message the lower node id starts the Ack timer and waits for acknowledgment message from the higher node id. If at the end of the Ack period the lower node id does not receive the acknowledgment message it declares itself as Cluster Head.

Hello Timer

On becoming cluster head a node needs to be aware of the status of its member nodes. This is necessary since sensor nodes die from time to time or new nodes might be deployed. In order to be aware of such topological changes a cluster head exchanges HELLO messages with its member nodes. A HELLO message is sent out every time the `Hello Timer` expires.

Neighbor Timer

Sensors die from time to time due to loss of power or due to other hardware problems. So the cluster head uses HELLO messages to know the status of the member nodes. The cluster head prunes away all the nodes that it did not hear from since the last time the Neighbor timer expired.

7.1.2 Functions

In this section we explain the implementation of various helper functions associated with the Clustering Agent.

FormPolyReqTimerexpired

```
Algorithm 1: Form Poly Inform message processing
  Input: None
  Output: None
  if received replies to FORM_POLY_REQ from at least 4 nodes then
      Try to form a valid polygon out of all the nodes in my neighbor list
      if can form a valid polygon then
          calculate how much new area is being covered by the currently selected
          polygon
          if new area > COVERAGE_FRACTION then
              compute the heuristic weight of the current polygon
              if heuristic_weight > old_heuristic_weight then
                  old_heuristic_weight = heuristic_weight
                  store the node ids of the current cluster
                  broad cast a FORM_POLY_INFORM message;
          else
              pick another set of four nodes
              if can find a new set then
                  goto line 3
              else
                  State = ORD_NODE;
      else
          Evaluate on every possible set of four nodes
```

As mentioned earlier the Polygon Request timer is used to solicit replies from the nodes that are interested in joining a new cluster. Algorithm 1 summarizes the processing done by a node to choose the best possible cluster out of its neighboring nodes. We keep evaluating every possible set of four nodes. The first step in the evaluation is to make sure that the cluster is a valid cluster. A cluster is considered to be valid if the length of the longest diagonal is smaller than the sensing range of the microphone. This ensures that every region enclosed by the convex hull of the resulting polygon is tracked. Once such a polygon is determined then we make sure that it incorporates a predefined amount of new area. New area is defined as the area that has not been covered by any polygon that has been created so far. Of all the polygons that meet these two requirements we select the one that is best in terms of heuristic weight. We experimented with two metrics. In the first case we simple choose the one that has the largest area without significance to the shape of the resulting polygon. In the next case we measure the degree of Irregularity of the resulting polygon and choose the one that is least irregular.

sendFormPolyReq

This helper function sends out a FORM_POLY_REQ message. This is a broadcast message. The sender then starts a timer. On receiving the message all the nodes that are interested in joining a cluster, it replies with a FORM_POLY_REPLY message.

PolyInformTimerexpired

This helper function deals with resolving conflict between nodes that intend to form identical clusters. A node that has moved to CHEAD_LOST state before the timer expires starts the process of forming a new polygon again. If it is in the state of POLY_INFORM_SENT, it then moves to CHEAD state. Algorithm 2 shows the pseudo code.

Algorithm 2: Form Polygon Inform Timer processing

```
Input: None
Output: None
if $State = CHEAD\_LOST$ then
    sendFormPolyReq
    return
if $State = POLY\_INFORM\_SENT$ then
    if $proportion \geq COVERAGE\_FRACTION$ then
        State = CHEAD
        broad cast a FORM_POLY_COMPLETE message
        store the newly formed cluster in local data base
    else
        State = CHEAD_LOST
        send out a fresh request to form new polygons
    if $State = CHEAD$ then
        Start a Hello timer for each of the member nodes
```

recvFormPolyInform

Every time a cluster head receives a FORM_POLYGON_INFORM message, it replies with a FORM_POLYGON_COMPLETE message. This message includes details of the member nodes of the cluster. A node that intends to form a cluster on receiving the FORM_POLYGON_INFORM message evaluates to determine if the cluster it intends to form is significantly different from the clusters that have already been formed. Two clusters are considered significantly different if they differ in area covered by a fraction of COVERAGE_FRACTION.

```
Algorithm 3: Form Poly Inform message processing
  Input: packet
  Output: None
  if $State = POLY\_INFORM\_SENT$ then
      evaluate how different my cluster is from the cluster information just received
      store the information in proportion
      if $proportion < COVERAGE\_FRACTION$ then
          if $iphrecv \rightarrow saddr() < my\_id\_$ then
              State = CHEAD_LOST
              $nblist \rightarrow setState(State)$
              sendFormPolyReq
          if $iphrecv \rightarrow saddr > my\_id\_$ then
              if $State = CHEAD\_LOST$ then
                  send out a STATUS_ACCEPTED message to $iphrecv \rightarrow saddr$
                  State = STATUS_ACCEPTED
```

Algorithm 3 summarizes the processing of an incoming FORM_POLY_INFORM message. If the current node is also in POLY_INFORM_SENT state then we need to evaluate if there is a tie. The condition (proportion < COVERAGE_FRACTION) indicates that there is a tie. This means that the cluster under consideration by the current node is not significantly different from the new cluster information that it has received. In such cases we need to ensure that the node with lower node id wins. So in case the current node's id is lower than that of the node id from which it received the message then the current node changes its state to CHEAD_LOST. On the hand if the current node's state is already in CHEAD_LOST then it replies with a STATUS_ACCEPTED message and changes its state to STATUS_ACCEPTED.

AckTimerexpired

Algorithm 4: Expiry of Ack Timer

```
Input: None
Output: None
if (State = ACK_WAIT) and (! AckRecvd) then
    Broad cast a FORM_POLY_COMPLETE message
    Store the new cluster formed in the local data base
    State = CHEAD;
```

Algorithm 4 summarizes the processing to be done on the expiry on Ack Timer. A node with lower id sends out a STATUS_ACCEPTED when it wants to let a node with higher node id become a cluster head. But in the event the higher node id does not confirm this message with ACK_STATUS_ACCEPTED message before the Ack Timer expires, then the current node sends out a FORM_POLY_COMPLETE message announcing its decision to become cluster head.

sendPolyDisband

A node is considered dead if it does not reply to three HELLO messages. In such situations the cluster head disbands the cluster. This is done by sending out a FORM_POLY_DISBAND message to the member nodes. Algorithm 5 gives the outline of the database update operation.

Algorithm 5: Form Poly Disband message

```
Input: failed
Output: None
if Cluster in the data base then
    Verify if the failed node is a member of this cluster
    if found then
        send out a FORM_POLY_DISBAND message to every member node
    delete every cluster that has failed node as its member node
```

recvPolyDisband

A member node on receiving FORM_POLY_DISBAND updates its membership information. Since a node could be part of more than one clusters formed by the same cluster head, it deletes its association with the cluster head only after the last cluster is disbanded. For this to be made possible, each member node keeps a counter that indicates the number of clusters it belongs to. Algorithm 6 summarizes the processing of FORM_POLY_DISBAND message. The message includes a full list of member nodes. This way the receiver can update its knowledge about the clusters in its vicinity. Finally, if the state of the node is either CHEAD or CHEAD_LOST or ORD_NODE then a fresh FORM_POLY_REQ is broadcast to form new cluster.

Algorithm 6: Form Poly Inform message processing

```
Input: Packet
Output: None
if member of the cluster disbanded then
    decrement the counter that tracks the number of clusters managed by the sender
    if counter = 0 then
        Cancel the Hello Timer

if Cluster info in local table then
    Remove the cluster information from the local table

if (State = CHEAD) or (State = CHEAD_LOST) or (State = ORD_NODE) then
    send out a fresh FORM_POLY_REQ message;
```

recvFormPolyComplete

A member node on receiving FORM_POLY_COMPLETE updates its membership information. A node could be part of more than one clusters formed by the same cluster head. To keep track of multiple associations with the same cluster head each member node keeps a counter that indicates the number of clusters it belongs to. Algorithm 7 summarizes the processing of FORM_POLY_COMPLETE message. The message includes a full list of member nodes. This way the receiver can update its knowledge about the clusters in its vicinity.

Algorithm 7: Form Poly Inform message processing

```
Input: Packet
Output: None
if member of newly created cluster then
    increment the counter that tracks the number of clusters managed by the sender
    Add the cluster information from the local table
    Start the Hello Timer
```

recvStatusAccepted

The issue of resolving ties is further complicated when more than two nodes are in a tie and at the same not every one in the group can hear every other member in the group. Algorithm 8 explains the resolution of ties in such complicated scenarios. On receiving a FORM_POLY_STATUS_ACCEPTED message a node changes it state to CHEAD and sends out FORM_POLY_COMPLETE message. On the other hand if the node is in STATUS_ACCEPTED_SENT state then it compares its own node id with that of the node that sent the message. If the id of the receiver is smaller than that of the sender then the receiver changes its state to CHEAD and sends out an ACK_STATUS_ACCEPTED message to the sender. It then broadcasts a FORM_POLY_COMPLETE message informing its member nodes about the creation of a new cluster.

Algorithm 8: Form Poly Inform message processing

```
Input: Packet
Output: None
if $State = CHEAD\_LOST$ then
    State = CHEAD

if $State = STATUS\_ACCEPTED\_SENT$ then
    if $iphrecv \rightarrow saddr() > my\_id\_$ then
        State = CHEAD
        send out an ACK_STATUS_ACCEPTED message
    else
        State = ACK_WAIT
        send out an ACK_STATUS_ACCEPTED message
        start the Ack timer

if $State = CHEAD$ then
    Broacast a FORM_POLY_COMPLETE message
    Store the new cluster information in the local table
```

7.2 Reliable Directed Diffusion

Implementation of Reliable Directed Diffusion protocol needed several modifications to Gradient Filter. Rather than merely storing the downstream node matching a gradient, the modified gradient filter stores more information that is gathered by the new messages that we added to the Diffusion protocol. Four new messages were added to diffusion protocol so as to enable the gathering of extra routing information necessary in the event link or node failure.

In the next section we will describe the implementation of Gradient filter in the Diffusion protocol and Section 7.2.1 we will explain the modifications necessary for the proposed Reliable Directed Diffusion.

7.2.1 Gradient Filter In Directed Diffusion

Gradient Filter stores routing information as a linked list and each entry. Every set of attributes has an associated routing entry in the routing table. The routing entry holds a gradient entry for every node from which it receives the same set of attributes. The gradient entry hold the address of the downstream node and port id in case the application is running locally. This allows for data aggregation based on attribute information. The gradient entry holds a flag that indicates if the gradient is reinforced or not. Data flows down only reinforced gradients.

Fig 7.1 shows the processing of an incoming packet by directed diffusion. Diffusion core determines the sequence of filter execution based on their respective priorities.

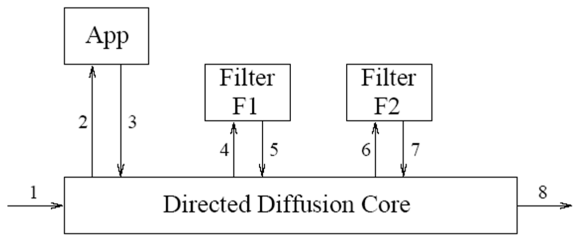

Figure 7.1: Message flow in directed diffusion

Routing In Directed Diffusion

Each message in Diffusion holds a packet number and a random id that is generated by the node that first creates the packet for communication. Directed diffusion core shown in Fig 7.1 is the first to receive an incoming packet. It creates a hash entry based on the the packet number and random id and stores the entry in a hash table. Duplicate packets are identified based on the information stored in the hash table and marked appropriately. The diffusion core eventually delivers the packet to Gradient filter. The gradient filter first extracts the attribute information and then maps this information to the appropriate routing entry. It then determines the next hop address based on the information stored in the routing entry. Gradient filter takes appropriate action based on the type of diffusion packet. If it is an INTEREST packet a new routing entry is created and the node address and port information stored as the case may be and the reinforcement flag is set to false. In case of POSITIVE_REINFORCEMENT the corresponding gradient entry flag is turned on. Similarly the flag is turned off on receiving NEGATIVE_REINFORCEMENT. If it is a DATA packet it is forwarded along every gradient that has been reinforced.

Modifications to Gradient Filter

In this section we explain the algorithm to process various messages introduced by Reliable Directed Diffusion. Algorithm 9 explains the processing of a Backup Positive Reinforcement message. Backup Positive Reinforcement message is a broadcast message. The message holds the source and destination nodes address of the link for which back up is needed. In this case, the source and destination node address would be the source and destination of the positive reinforcement message for which the back up message was generated. For every backup Positive Reinforcement message that a node receives, it records the message attributes in a vector form. This implies that the node intends to play the role of back up node for the set attributes. In addition to the attributes it also records the source and destination address of the link for which it intends to play the role of back up node.

Algorithm 9: Backup Positive Reinforcement message processing

```
Input: Packet
Output: None
Extract the reinforcement attributes
if ! reinforcementAttr then
    Ërror: Received an invalid Back up Positive Reinforcement message!̈; return
Remove the reinforcement attributes
Look for the back up Routing Entry that matches the attributes in the Back up
Positive Reinforcement message
if ! routingEntry then
    Create a new routing entry for this data type
    Create a new Link entry for this data type
    update the routing entry with the new link entry
    update the back up routing table
```

A new routing table name ”Backup routing table‘ ”’ is created in order to store the information provided by back up messages.

```
class TppRoutingEntry {
  AgentList agents;
  GradientList gradients;
  AttributeList attrList;
  DataNeighborList dataNeighbors;
  LinkList llist;

  Helper Functions;
}
```

- `agents` - agents represents is a linked list of all the agents that are interested in the DATA matching the attributes provided in the attrList.
- `gradients` - Stores the Gradient information for which the current back up request message has been received.
- `attrList` - Holds the list of all the attributes for which Backup has been requested.
- `dataNeighbors` - Holds the information about the neighboring node Ids.
- `llist` - Hold the more details about the link for which the current node needs to back up.

Every time a node receives a back up message, it needs to record the certain link properties for which it intends to back up. This information is stored in a linked list. Each element of the list is defined as class LinkEntry.

```
class LinkEntry{
public:
  int32_t Sid;
  int32_t Did;
  int32_t failAttempts;
  int32_t type;
  int32_t srcNode;
  int32_t srcNodeType;
  struct timeval tmv;
};
```

- `Sid` - Represents the Source ID of the link for which a back up node needs to be identified.
- `Did` - Represents the Destination ID of the link for which a back up node needs to be identified.
- `failAttempts` - The number of times a DATA packet could not be transmitted along the link. The counter is incremented for every failure and decremented for every success.
- `type` - Indicates the type of message, could be either Backup Positive Reinforcement or Backup Data message.
- `srcNode` - The source node that generated the backup message.
- `srcNodeType` - The nature of the source node. The source node that generated the backup message could be either a backup node or main node.
- `tmv` - record the time at which the message was received.

The basic directed diffusion stores the gradient information. However to implement the reliable diffusion protocol we had to make a few changes. The modified Gradient Entry class is defined below.

```
class GradientEntry {
public:

  BkpList bkplist;
  int32_t node_addr_;
  int32_t fail_attempts_;
  struct timeval tmv;
  bool reinforced_;
};
```

- `bkplist` - Linked list of all nodes that have agreed to back up this gradient.
- `node_addr_` - Address of the downstream node.
- `fail_attempts_` - Number of consecutive attempts that the link has failed.
- `tmv` - record the time at which the message was received.
- `reinforced_` - Is a boolean variable that indicates whether the gradient is reinforced or not.

Algorithm 10 explains the processing of a Backup data message. Backup data message is a broadcast message. The message holds the source and destination nodes address of the link for which back up is needed. In this case, the source and destination node address would be the source and destination of the DATA message for which the back up message was generated. For every backup DATA message that a node receives, it records the message attributes in a vector form. This implies that the node intends to play the role of back up node for the set attributes. In addition to the attributes it also records the source and destination address of the link for which it intends to play the role of back up node. If the back up node had earlier recorded a back up positive reinforcement message for the same link, it concludes that it can back up the said link and declares itself to be a BACK_UP node. It then sends out a REPLY message to the node that had earlier broadcasted the BACKUP_DATA message. Algorithm 12 explains the processing necessary for BACKUP_DATA reply message.

```
Algorithm 10: Backup Data message processing
  Input: Packet
  Output: None
  if If Received the message from a BACKUP_NODE then
      Look for the back up Routing Entry that matches the attributes in the Back up
      Positive Reinforcement message
      if ! routingEntry then
          Create a new routing entry for this data type
          Create a new Link entry for this data type
          update the routing entry with the new link entry
          update the back up routing table

      while $link\_itr1 \neq bkproutingEntry \rightarrow Reqllist.end$ do
          if (linkEntry → srcNodeType = BKPUP_NODE) and
          (linkEntry → type = 2) and (linkEntry → Sid = msg → bkpup ) then
              bkpupnode = $linkEntry1 \rightarrow srcNode$
              bkpupfor = $msg \rightarrow lastHop$
              replyto = $msg \rightarrow lastHop$
              bkpup1 = $msg \rightarrow bkpup1$
              bkpup2 = $msg \rightarrow bkpup2$
          end
          create a link entry with bkpupnode, bkpupfor, replyto, bkpup1 and bkpup2
          Store the link information as one of the links that is being backed up
          Add the id of bkpup node to the list of nodes that are being backed
          Send BACKUP_DATA_REPLY message to lastHop
      end
  else
      while routingEntry do
          while linkEntry do
              if (linkEntry → Did = msg → bkpup2) and (linkEntry → Sid =
              msg → bkpup1) and (linkEntry → type = 1) then
                  flag = true
                  break
              end
          end
      end
      if flag then
          Add $msg \rightarrow msgAttrVec$ to the list of attributes that are being backed
          create a link entry with $msg \rightarrow lastHop$, bkpupfor, replyto, $msg \rightarrow bkpup1$
          and $msg \rightarrow bkpup2$
          Store the link information as one of the links that is being backed up
          Send BACKUP_DATA_REPLY message to lastHop
```

Algorithm 11: Backup Data message processing...Contd

```
Input: Packet
Output: None
$msg \rightarrow hops --$
if $msg \rightarrow hops > 0$ then
    Re broad cast the BACKUP_DATA message
```

Algorithm 12: Backup Data Reply message processing

```
Input: Packet
Output: None
if (Am a back up node) and (Received message from BACK_UP node) then
    while routingEntry do
        while gradEntry do
            Add $msg \rightarrow lastHop$ to backup node list
            Sort the back up node list by geographical distance
        end
    end
else
    Iam a Main node
  while routingEntry do
    while gradEntry do
        Add $msg \rightarrow lastHop$ to backup node list
        Sort the back up node list by geographical distance
    end
end
```

Algorithm 13 explains the process for a MAIN node to reroute DATA packets around a link and node failure. If DATA packets are rerouted through BACKUP nodes for GF_RETRIES then the MAIN node that receives the REROUTE DATA packet concludes that the failure is permanent and hence sends out POSITIVE REINFORCEMENT message back to the node that originated the REROUTE DATA packet. As the POSITIVE REINFORCEMENT message passes through a series of Back up nodes they reconfigure themselves as main nodes and remain until they receive a NEGATIVE REINFORCEMENT message.

Algorithm 13: Reroute Data message processing

```
Input: Packet
Output: None
if $(msg \rightarrow failedNode \neq -1)$ and $(msg \rightarrow failAttempts \geq GF\_RETRIES)$ then
    processRerouteData(msg);
end
if $msg \rightarrow nodeType = MAIN\_NODE$ then
    if $msg \rightarrow failAttempts < GF\_RETRIES$ then
        $msg \rightarrow nextHop = msg \rightarrow bkpup1$
    else
        $msg \rightarrow nextHop =$ nexthop
    $msg \rightarrow nodeType =$ BKP_NODE
    $msg \rightarrow msgType =$ REROUTE_DATA
    $msg \rightarrow bkpup1 = msg \rightarrow nextHop$
    $msg \rightarrow bkpup2 = msg \rightarrow lastHop$
    $msg \rightarrow xmitFailure =$ GradXmitFailedCallback
    $msg \rightarrow xmitFailureData =$ (void*) this
    Send message
else
    while routingEntry do
        while linkEntry do
            if $(linkEntry \rightarrow type = 2)$ and $(linkEntry \rightarrow srcNodeType = MAIN\_NODE)$ then
                $msg \rightarrow nextHop = linkEntry \rightarrow Did$
                $msg \rightarrow nodeType =$ BKP_NODE
                $msg \rightarrow msgType =$ REROUTE_DATA
                $msg \rightarrow bkpup1 = msg \rightarrow nextHop$
                $msg \rightarrow bkpup2 = linkEntry \rightarrow Sid$
                $msg \rightarrow xmitFailure =$ GradXmitFailedCallback
                $msg \rightarrow xmitFailureData =$ (void*) this
                Send message
            end
        end
    end
```

Algorithm 14 explains the processing of packet that could not be successfully transmitted. We rely on the MAC layer ACK mechanism to determine the success or failure of a transmitted packet. On failure, the MAC layer hands over the packet to the Network layer through a callback mechanism. The network layer then looks through the list of back up nodes and determines the next unused back up node. If all the back up nodes have been attempted then we simply broadcast the DATA message and hope that one of the downstream back up nodes receives the message.

Algorithm 14: Failed packet processing

```
Input: Packet
Output: None
if ($msg \rightarrow msgType$ = DATA) or ($msg \rightarrow msgType$ = REROUTE_DATA) then
    if at a BACKUP_NODE then
        while routingEntry do
            while linkEntry do
                if ($linkEntry \rightarrow Did = msg \rightarrow nextHop$) and ($linkEntry \rightarrow type$
                = 2) then
                    if Message in Error then
                        $linkEntry \rightarrow failAttempts++$
                    end
                    else
                        $linkEntry \rightarrow failAttempts--$
                    end
                    if $linkEntry \rightarrow failAttempts \geq$ GF_RETRIES then
                        $linkEntry \rightarrow failAttempts$ = GF_RETRIES
                    end
                    else
                        $linkEntry \rightarrow failAttempts \leq 0$
                    end
                    $linkEntry \rightarrow failAttempts = 0$
                    break;
                end
            end
            if attempts $\neq$ -1 then
                break
            end
        end
        bkpup = findbkpnodeforbkp(msg, $msg \rightarrow nextHop$)
        if $msg \rightarrow nextHop = bkpup$ then
            $msg \rightarrow nextHop$ = BROADCAST_ADDR
        end
        else
            $msg \rightarrow nextHop$ = bkpup
        end
        $msg \rightarrow msgType$ = REROUTE_DATA
        $msg \rightarrow nodeType$ = BKP_NODE
        $msg \rightarrow xmitFailure$ = GradXmitFailedCallback
        $msg \rightarrow xmitFailureData$ = (void*) this
        if Message in error then
            send message
        end
    end
end
```

Algorithm 15: Failed packet processing...Contd

```
Input: Packet
Output: None
else
    attempts = -1;  while routingEntry do
        while gradientEntry do
            if gradientEntry → nodeAddr = msg → nextHop then
                if Message in error then
                    gradientEntry → failAttempts ++;
                end
                else
                    gradientEntry → failAttempts −−;
                end
                attempts = gradientEntry → failAttempts
                if gradientEntry → failAttempts ≥ GF_RETRIES then
                    gradientEntry → failAttempts = GF_RETRIES
                end
                else if gradientEntry → failAttempts ≤ 0 then
                    gradientEntry → failAttempts = 0
                end
                break;
            end
        end
        if attempts ≠ -1 then
            break;
        end
    end
    bkpup = findbkpnodeforMain(msg);
    msg → failedNode = msg → nextHop
    msg → msgType = REROUTE_DATA
    msg → nodeType = MAIN_NODE; msg → failAttempts = attempts;
    msg → xmitFailure = GradXmitFailedCallback
    msg → xmitFailureData = (void*) this
    msg → nextHop = bkpup; if (Message in Error) and (msg → nextHop ≠ -1)
    then
        Send the message
    end
end
```

Chapter 8

Performance Evaluation

8.1 Performance of Clustering Algorithm

8.1.1 Experimental Setup

The clustering algorithm was implemented in ns-2.27 [57]. In all, five different deployment topologies were evaluated, i.e. different pentagonal tessellation algorithms, grid deployment and random deployment.

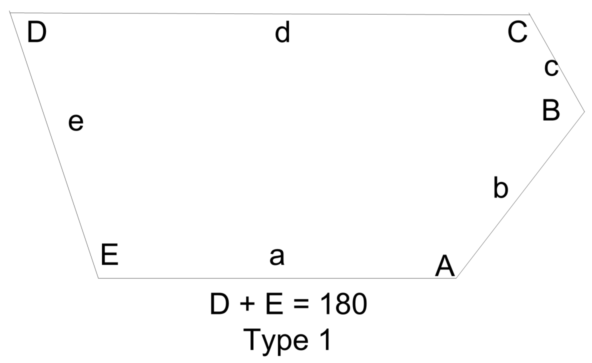

Figure 8.1: Pentagonal tessellations.

Pentagonal tilings were analyzed since it is possible to have non-overlapping clusters of five sensors and hence assure that each part of the field can be monitored by at least one cluster. In an ideal case where each cluster can monitor the region that it encloses with a probability of 1. By having one of the five nodes of each pentagon to be a cluster head and the other nodes as its member nodes, the entire field can be monitored with a minimum

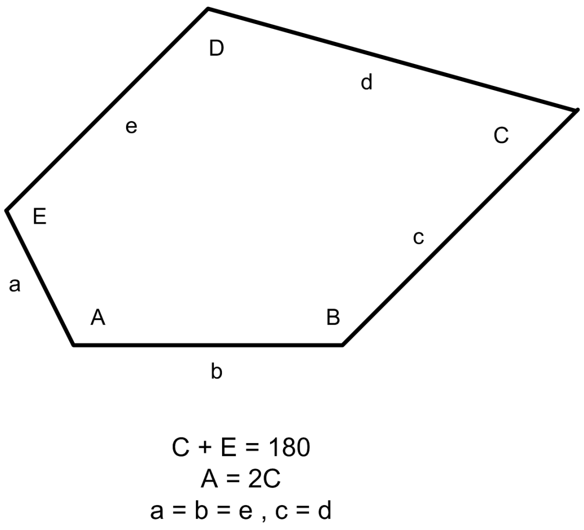

Figure 8.2: Pentagonal tessellations.

number of clusters. In all, 14 different types of convex pentagons [58] can tile a plane but we evaluated only two of them. Figures 8.1, 8.2 and 8.3 shows the geometrical properties of the three of the possible 14 tilings.

In tessellation of Type I, 248 sensors were deployed in a rectangular field of 500 * 1000m. In case of Type II, a total of 245 nodes were deployed in a rectangular field of 1000*1000m. Finally in case of type III tessellation, a total of 264 nodes were deployed in a rectangular field of 1000*1000m. In case of grid deployment in which 300 nodes were laid in 20 * 15 matrix and separated by 40m. Finally, in case of random deployment 300 sensors uniformly deployed over a 760*560 m field. The communication range of the sensor was set to twice the range of the acoustic signal.

Figures 8.4, 8.5 and 8.6 show a snapshot of the node deployment using Type 1, Type 2 and Type 3 pentagonal tiling.

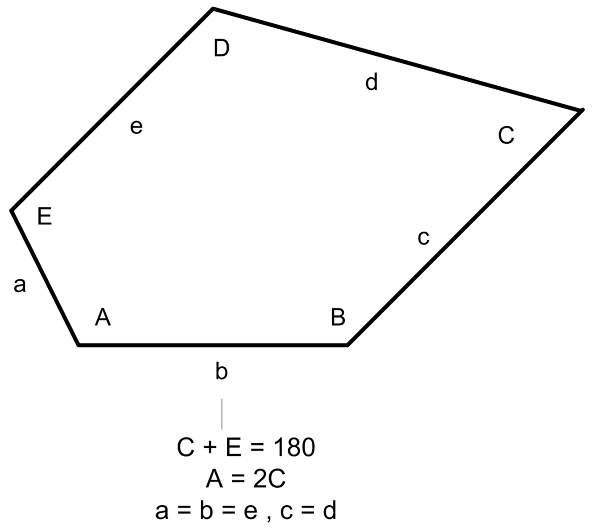

Figure 8.3: Pentagonal tessellations.

The parameters of interest are the total number of polygons created, Degree of Coverage, Average Probability of Detection, Protocol Overhead, Breach Weight and Support Weight. We compare two different techniques of forming polygons. In the first technique each node tries to form a pentagon that is most regular. The degree of irregularity is measured by Equation 4.1. We call this technique mathematical approach. In the other technique each sensor forms a pentagon out of nodes that are closest to it. We call this non-mathematical approach.

Figures 8.7, 8.8, 8.9 and 8.10 compare the performance of mathematical and non mathematical techniques while using the single cluster technique. In single cluster technique each sensor is allowed to form and manage only one cluster.

Figure 8.7 compares the number of polygons the mathematical and non-mathematical technique would form for the five different node deployment scenarios. In case of pentagonal

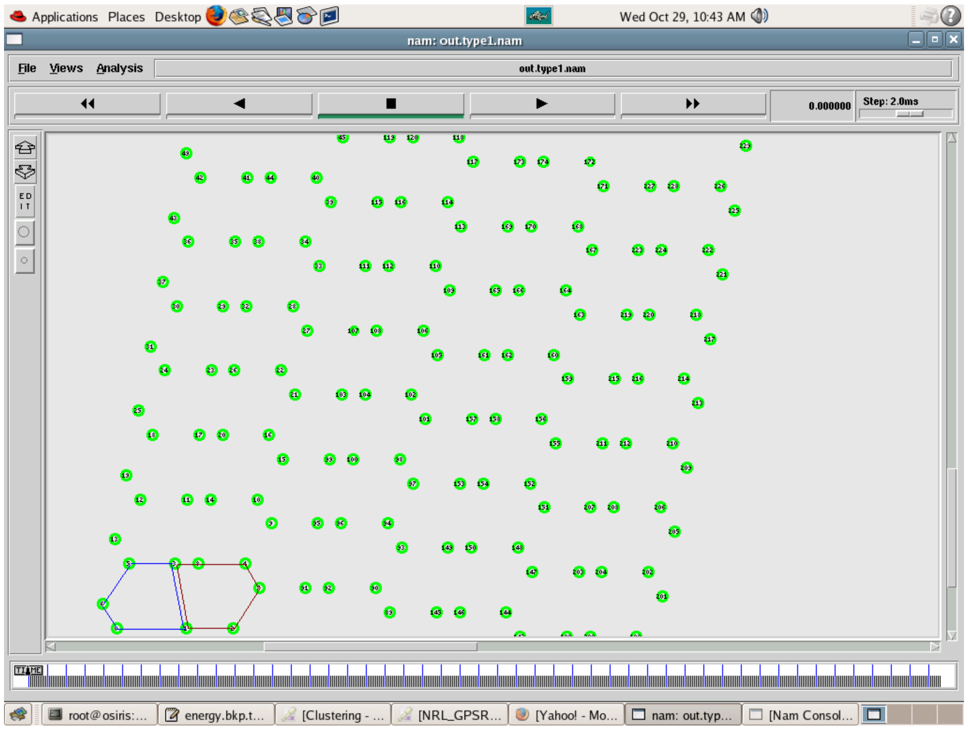

Figure 8.4: Node deployment using Type 1 pentagonal tiling.

tessellation the number of polygons required to completely track the entire field is equal to the number of pentagons in the tessellation.

The number of polygons identified by both the mathematical and non mathematical technique is close to the number of polygons identified in theory to entirely cover the field.

But from Figures 8.8, 8.9 and 8.10 we can notice that when a node forms a polygon using the mathematical model it achieves greater degree of coverage and average probability of detection when compared to nonmathematical model. The performance of the mathematical model is very much closer to the theory.

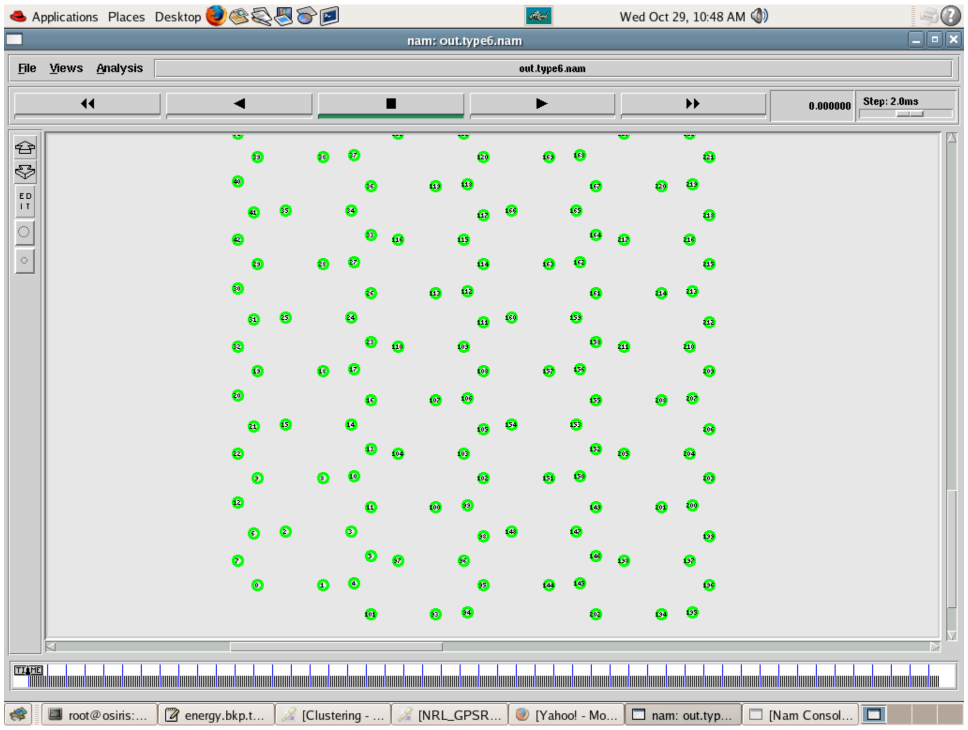

Figure 8.5: Node deployment using Type 2 pentagonal tiling.

Since both mathematical and non-mathematical model form identical number of polygons and hence the energy consumed by both the techniques is identical. This can observed in Figure 8.10.

Figure 8.11 compares the breach path, which is a path from initial location I to final location F with its smallest weighted edge being as large as possible. Lower breach weight represents better coverage of the sensor field. It can be seen that by using mathematical approach we can achieve lower breach weight. Similarly, Figure 8.12 compares the support weight achieved by using mathematical approach and non-mathematical approach. Unlike breach weight which evaluates the worst case coverage, support weight represents the best

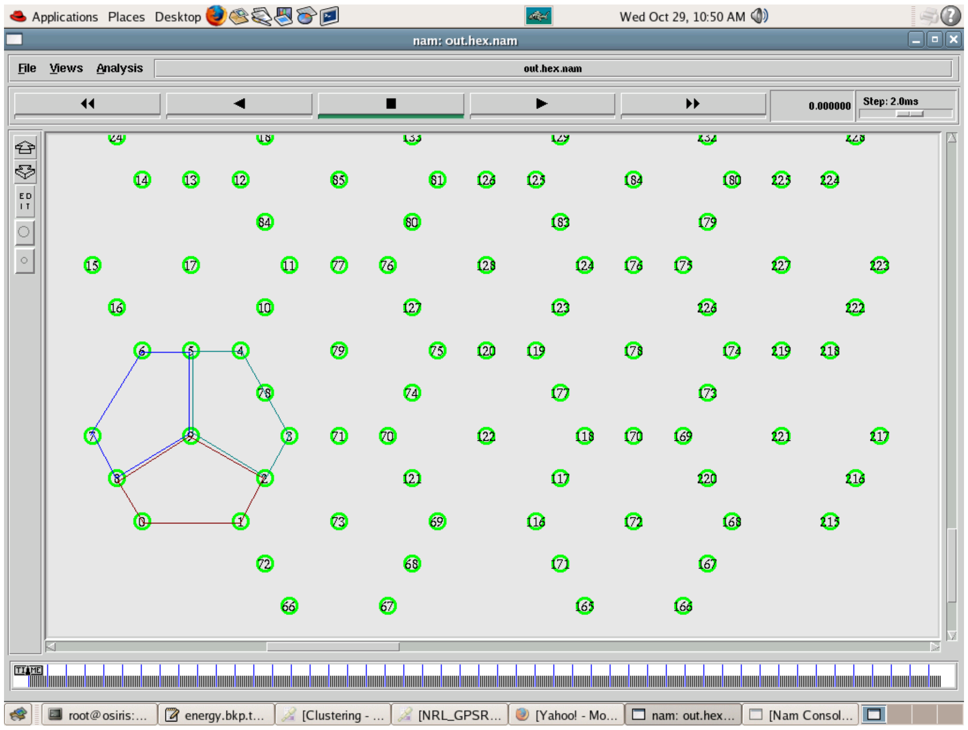

Figure 8.6: Node deployment using Type 3 pentagonal tiling.

case coverage. A lower value represents better coverage. Once again it can be seen that mathematical approach achieves better support weight when compared to non-mathematical approach.

From Figures 8.8 and 8.9 we can conclude that forming more regular pentagons improves the degree of coverage and average probability of detection.

But because of the inaccuracy of the CPA algorithm in identifying the CPA time accurately, with each node forming only one cluster the probability of detection is below the theoretical best of 1.0. We can achieve this theoretical best by allowing each node to form more than one cluster.

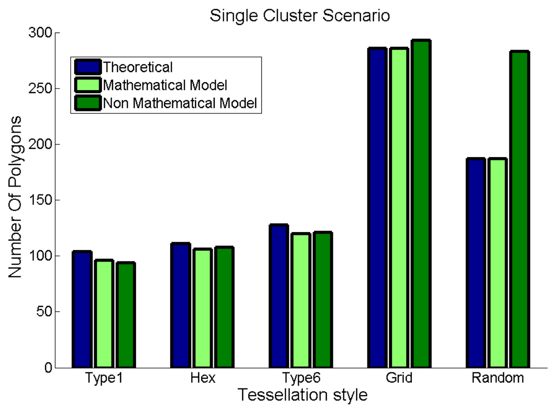

Figure 8.7: Number of clusters formed by Mathematical and non Mathematical techniques while using single cluster technique in comparison to theoretical technique.

To analyze the improvement in probability of detection achieved by having multiple clusters, we need to have an error model for CPA algorithm. Figure 8.13 represents a hypothetical error model for the CPA algorithm.

Figures 8.14, 8.15, 8.16 and 8.17 compare the performance of single cluster and multi-cluster algorithms while using the mathematical approach in forming a cluster.

From Figure 8.14 it can been seen that the number of polygons formed by the multi-cluster approach is close to twice the number of polygons formed by single cluster approach. This is more evident when the nodes are deployed using Type I, Type II and Type III pentagonal tessellations.

From Figure 8.15 it can be seen that the degree of coverage achieved by single cluster and multi-cluster approaches is close to 100 percent, but the average probability of detection is greatly improved by allowing each cluster to form more than one cluster. This improvement in average probability of detection is evident in Figure 8.16. Multi-cluster approach

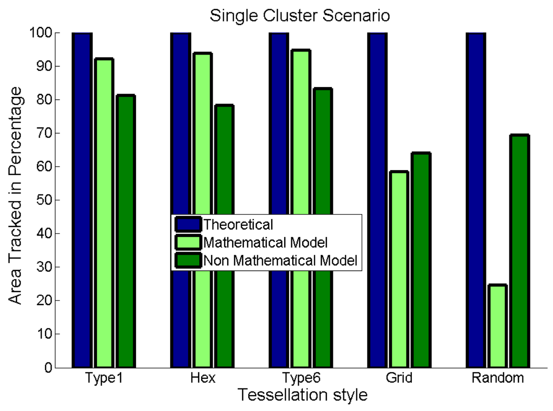

Figure 8.8: Maximum Degree of Coverage achieved by Mathematical and non Mathematical techniques while using single cluster technique in comparison to theoretical technique.

achieves a higher probability of detection because of the overlapping nature of the clusters. This means that we have more than one cluster track the same region.

The improvement in average probability of detection when using multi-cluster approach comes with a cost. More number of clusters means more protocol packets to manage and the clusters and hence more energy is consumed. Figure 8.17 compares the average node energy consumed by both single cluster and multi-cluster approach. Energy being a scare resource in sensor networks, it is important to strike a balance between the average probability of detection and the average energy consumed by each node in the network.

Figures 8.18 and 8.19 compare the improvement in breach and support weight achieved by forming multiple clusters instead of single cluster respectively. In both cases the clusters are formed using mathematical approach.

Whenever a node dies the cluster heads of all the clusters that the dead node belongs to are dissolved and a race for forming new clusters begins. Also since a node, can be a cluster head and at the same time be a member node of another cluster, the dissolution of one

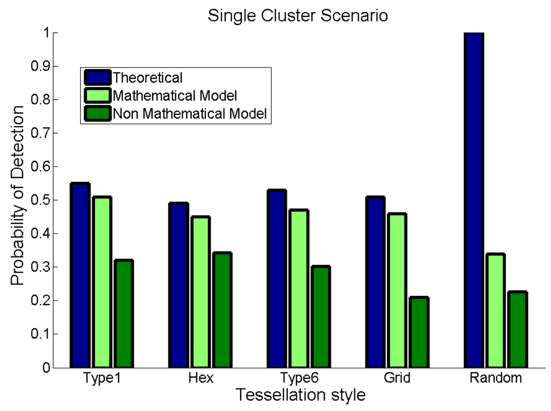

Figure 8.9: Average Probability of Detection attained by Mathematical and non Mathematical techniques while using single cluster technique in comparison to theoretical technique.

cluster could lead to a ripple effect. A larger ripple means a larger number of clusters have to be reorganized and hence more protocol overhead. The extent of the ripple is dependent on the geographical location of the node and its relationship to other nodes. In an ideal case the clustering algorithm should be able to re organize the clusters with minimum energy and at the same time achieve the best possible degree of coverage and average probability of detection. Figures 8.20, 8.21, 8.22 and 8.23 analyze the ability of the clustering algorithm to reconfigure the clusters in the event of node failures. In each simulation set up about 5% of the nodes were turned off. To simulate the failures necessary to test the reconfigurability of the clustering algorithm, we generated failure scenarios consisting of a series of node failures over the course of the simulation. The node failures were drawn from an exponential distribution to model failures during the normal useful-life phase of the system [59]. The mean time between failures for the exponential distribution was β where $\beta \in \{2.5, 5, 10, 15, 25\}$.

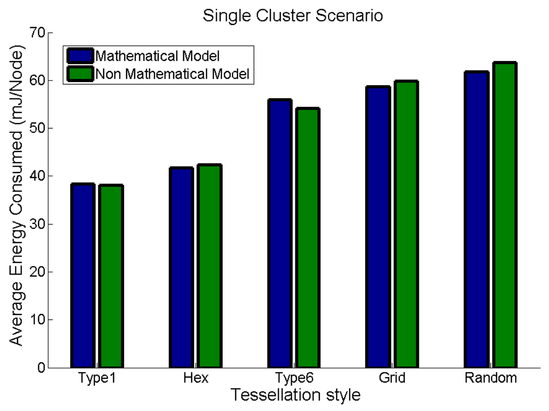

Figure 8.10: Total energy consumed by Mathematical and non Mathematical techniques while using single cluster technique.

From Figure 8.20 it can be seen that there is only a small loss in the total number of clusters. This implies that a vast number of new clusters were successfully formed to replace the clusters in which one of the nodes failed.

Figures 8.22 and 8.21 compare the the average probability of detection and degree of coverage before and after reconfiguration. After reconfiguring the clusters there is only a small decrease in the degree of coverage and average probability of detection. Hence, it can be concluded that the clustering algorithm can successfully reconfigure without great loss in the average probability of detection and degree of coverage.

To make reconfiguration possible the nodes need to exchange information and this consumes energy. Figure 8.23 compares the protocol overhead incurred to re-organize the clusters. It can be seen that the amount of energy consumed in re-organizing the clusters is minimal.

Just as the reconfiguring clusters in the event of node failure affects parameters like coverage, probability of detection and energy, it also affects the breach weight and support

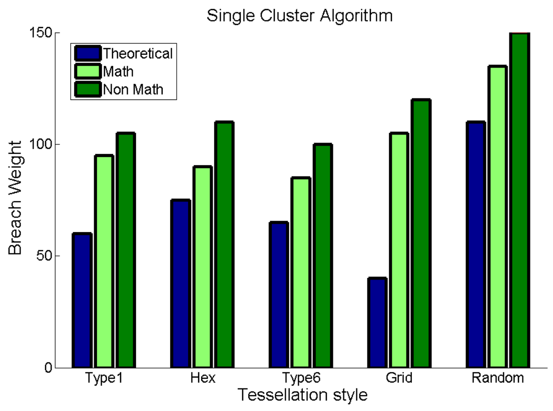

Figure 8.11: Breach weight of Mathematical and non Mathematical techniques while using single cluster technique.

weight achieved by the clustering algorithm. Figures 8.24 and 8.25 show how cluster reconfiguration affects the breach and support weight. It can be concluded that the algorithm is able to reconfigure the clusters without significantly affecting the breach and support weights.

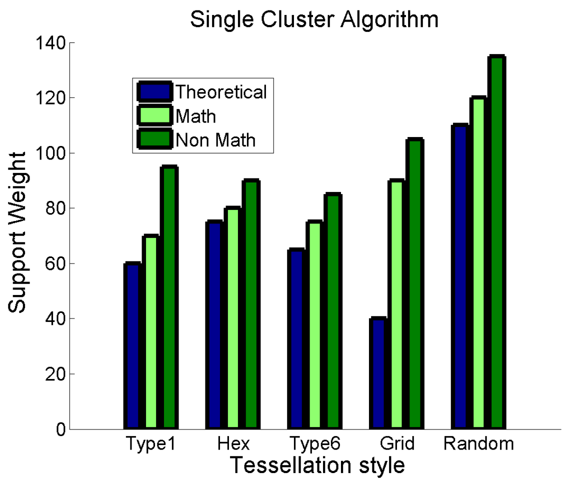

Figure 8.12: Support Weight of Mathematical and non Mathematical techniques while using single cluster technique.

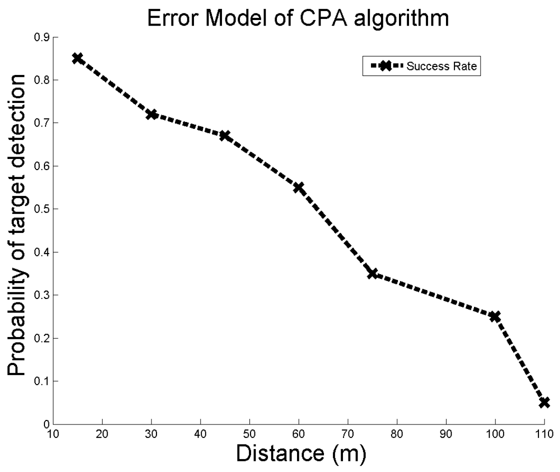

Figure 8.13: Error Model representing the probability of target detection by the CPA algorithm with distance.

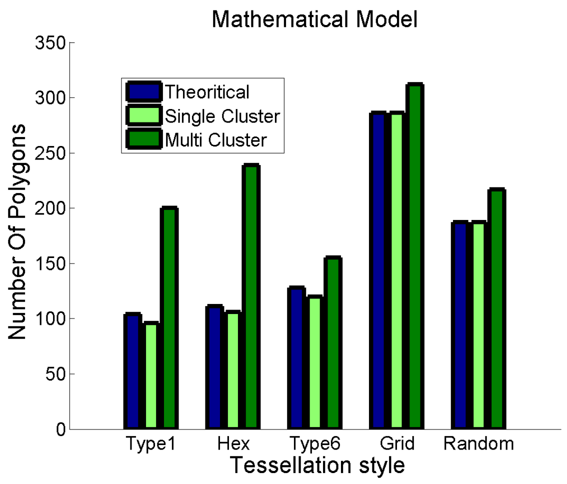

Figure 8.14: Number of clusters formed by Single cluster and Multi-Cluster algorithms while using Mathematical approach in forming a cluster.

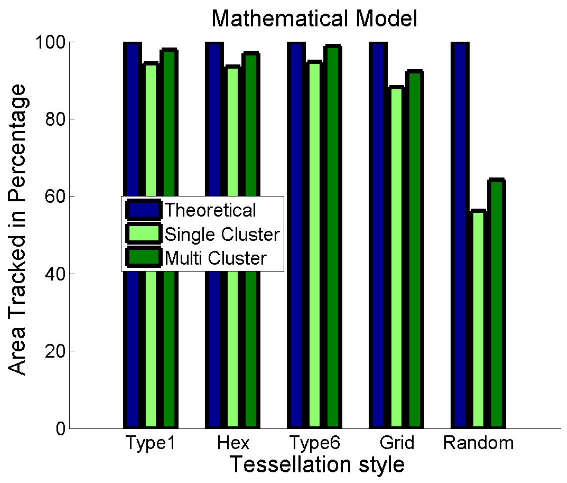

Figure 8.15: Maximum Degree of Coverage achieved by Single cluster and Multi-Cluster algorithms while using Mathematical approach in forming a cluster.

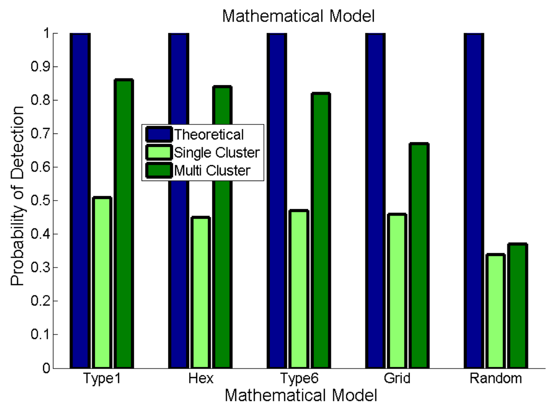

Figure 8.16: Average Probability of Detection achieved by Single cluster and Multi-Cluster algorithms while using Mathematical approach in forming a cluster.

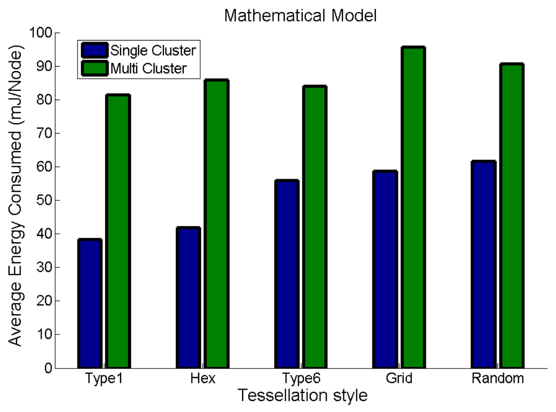

Figure 8.17: Total energy consumed by Single cluster and Multi-Cluster algorithms while using Mathematical approach in forming a cluster.

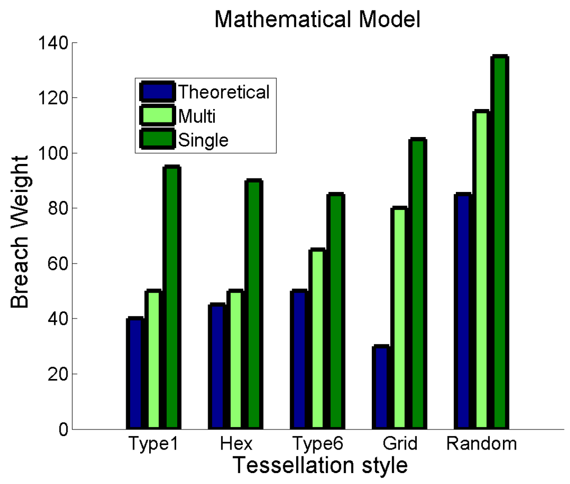

Figure 8.18: Breach Weight of Single cluster and Multi-Cluster algorithms while using Mathematical approach in forming a cluster.

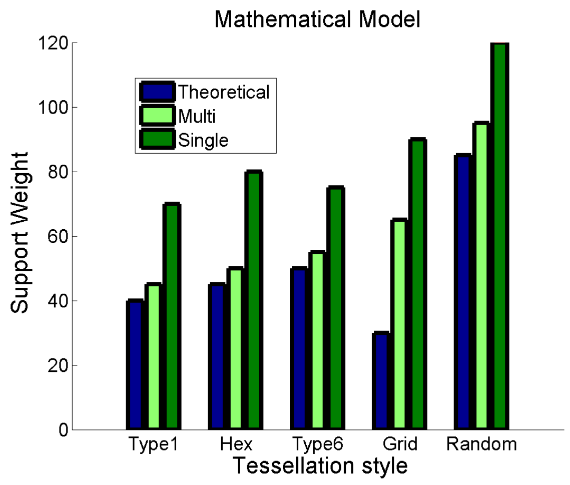

Figure 8.19: Support Weight of Single cluster and Multi-Cluster algorithms while using Mathematical approach in forming a cluster.

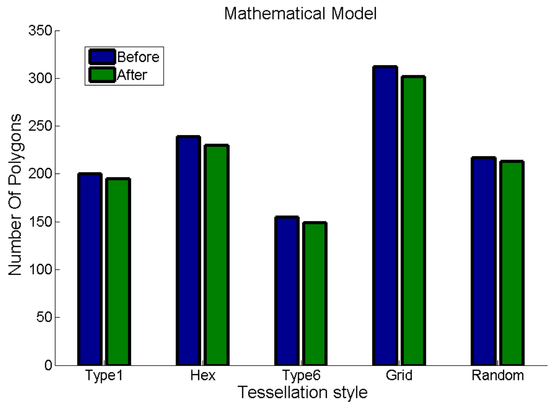

Figure 8.20: Number of clusters formed by Multi-Cluster algorithm while using Mathematical approach before and after some of the nodes die.

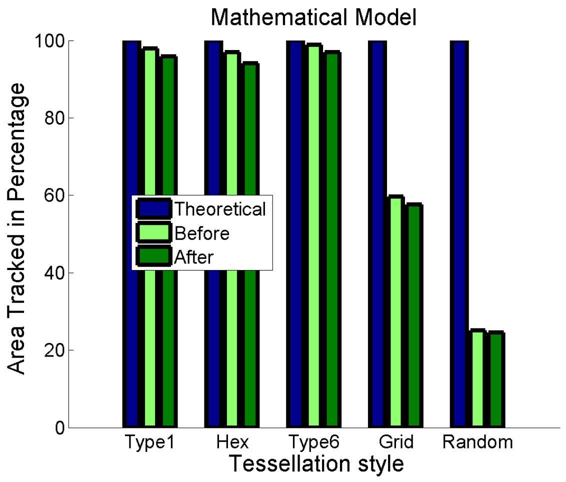

Figure 8.21: Maximum Degree of Coverage achieved by Multi-Cluster algorithm while using Mathematical approach before and after some of the nodes die.

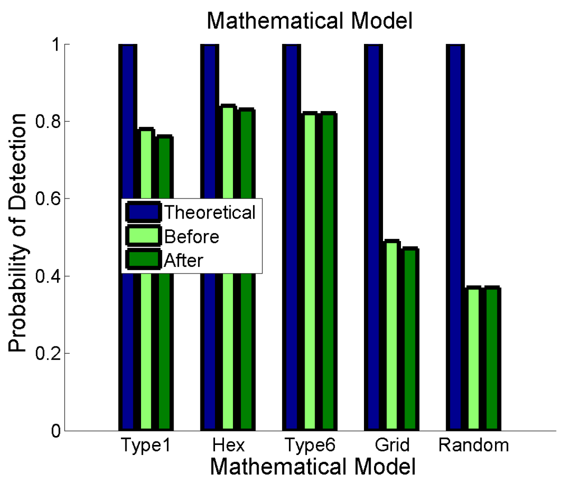

Figure 8.22: Average Probability of Detection achieved by Multi-Cluster algorithm while using Mathematical approach before and after some of the nodes die.

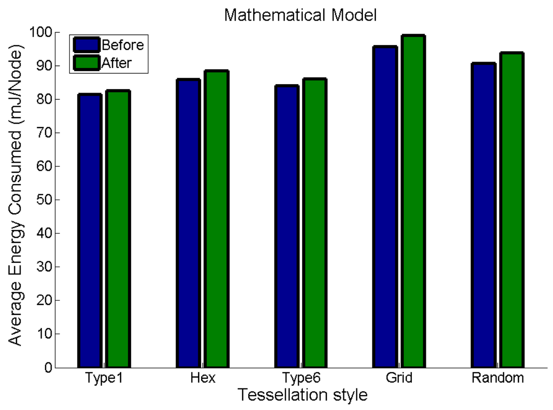

Figure 8.23: Total energy consumed by Multi-Cluster algorithm while using Mathematical approach before and after some of the nodes die.

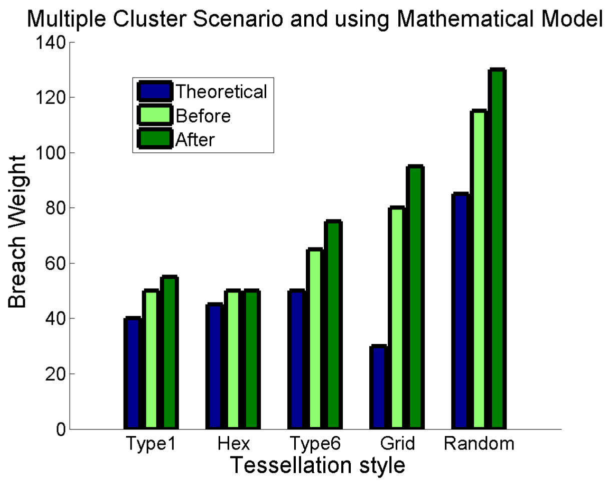

Figure 8.24: Breach Weight of Multi-Cluster algorithm while using Mathematical approach before and after some of the nodes die.

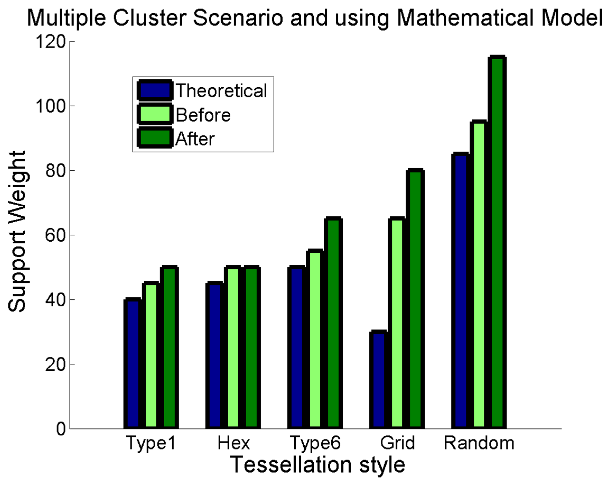

Figure 8.25: Support Weight of Multi-Cluster algorithm while using Mathematical approach before and after some of the nodes die.

8.2 Reliable Directed Diffusion

We compare the performance of Reliable Directed Diffusion, RMST and basic directed diffusion, in terms of delivery ratio, end-to-end delay and energy consumed (measured in mJ/pkt).

8.2.1 Experimental Setup

Ns-2.29 [57] was used for simulations. Two topologies were tested: grid and random. In the grid topology, 100 nodes were positioned in 10 rows and 10 columns. While in the random topology 150 nodes were uniformly distributed in a 670 * 670 meter region. In both topologies, the source and sink were chosen such that the average path length was 10-15 hops. The reported results are based on the average of 22 runs.

In the simulations we measured the energy consumed to deliver a packet (mJ/pkt), hence the energy model is significant. We used the IEEE 802.11 implementation of ns-2. For such a CSMA MAC, each node must listen to the channel continuously. Hence, the idle power is comparable to that of receive power. We set the Tx power to 0.660W, Rx power to 0.395W and idle power to 0.390W.

The wireless losses are simulated using the model based on the implementation provided in [60]. The error model evaluates the received signal strength of a packet by considering both noise and interference of neighboring nodes. The bit error rate is calculated from the received signal strength by Formula 8.1 [61]

$$P_b = 0.5 \cdot erfc\left(\sqrt{\frac{P_r \cdot W}{N \cdot f}}\right) \tag{8.1}$$

where P_r is the received power, W is the channel bandwidth, N is the noise power, f is the transmission bit rate, and $erfc()$ is the complementary error function. The frame error rate is then calculated based on the length of the packet received.

A constant bit rate (CBR) application was used for application layer traffic. The data rates used were 10, 15, 25, 50, 70, 80, and 100 packets per second. The size of each data packet is 200 bytes. The simulation was run for 100 seconds.

8.2.2 Results and Discussions

Figure 8.32 compares the delivery ratio of reliable directed diffusion, RMST and Directed Diffusion for the grid scenario. The delivery ratio was evaluated for different data rates. The minimum data rate was 10 packets per second (PPS) and the maximum data rate was 100 PPS. Figure 8.35 compares the delivery ratio of reliable directed diffusion, RMST and Directed Diffusion for the random scenario. In both cases, the network is saturated when the data rate is between 20 PPS and 100 PPS. In the saturated network (congested), the losses are mostly due to collisions rather than wireless errors. Under saturated conditions, reliable diffusion shows a small improvement. This is because the additional protocol packets introduced by reliable diffusion worsen the congestion and, hence, reduce the gain in delivery ratio. When the data rate is less than 20 PPS the network experiences no congestion and hence the delivery ratio increases for both reliable diffusion and directed diffusion. The delivery ratio saturates at slight below 80% in the case of diffusion while it saturates at about 96% for reliable diffusion. Under low data rate conditions, most of the packet losses are wireless in nature thus RDD can successfully recover them. RDD improves the delivery ratio by about 20%. In the random scenario, directed diffusion achieves a maximum delivery ratio of about 72% whereas reliable directed diffusion achieves a maximum delivery ratio of 96%. Thus, in the random scenario, RDD gives a relative improvement of 35%.

Figures 8.33 and 8.36 compare the average energy (measured as mJ/pkt) consumed to deliver a data packet for grid and random topologies respectively. We see that RDD consumes only about 2% more energy more than DD. As the data rate increases, the average energy consumed to delivery a packet decreases. This is because the protocol overhead involved to setup a path between source and sink does not depend on the data rate. Thus, at low data rates the network is under-utilized. The energy consumption reaches a minimum at about 40 PPS. Beyond this, the energy consumption marginally increases because of network congestion past this threshold. Network congestion leads to packet collisions at the MAC layer and, as a result, more data transmissions are needed to deliver a packet. The network congestion is made even worse by the extra packets generated by RDD to build backup nodes. The confirmation messages from backup nodes generated in response to backup requests from main nodes may be lost because of collisions and this reduces overall delivery ratio while the energy consumed remains the same. Thus, under congestion, reliable diffusion consumes 6% more energy than basic diffusion.

In the grid topology, RDD achieves higher delivery ratio but consumes about 6% more energy. This is because delivering packets to the next hop via backup nodes does not necessarily mean shorter hops, but may be the very nature of grid layout delivering packets to the backup nodes may travel along the diagonal of a grid rather than the edge of the grid. This leads to more energy consumption when compared to random deployment. In the random topology, the nodes are uniformly deployed so RDD is more often able to find a backup node (in the event of failure) which offers a shorter hop distance. In the random topology Figure 8.28, the energy consumed to build backup nodes is 11% more productive in delivering data packets than in the grid layout. As a result, the protocol overhead in terms of energy consumed per packet delivered is reduced.

Figures 8.34 and 8.37 compare the end-to-end delay for both techniques for both grid and random topologies. There is no significant difference in the end-to-end delay of standard directed diffusion and RDD for low data rates. This is because reliable diffusion packets only take an extra hop in case of routing failures. Under low data rate conditions (below 20 PPS), most of the losses are wireless in nature and, on average, the packets take only couple of extra hops to reach the destination in RDD. Under heavy load, the extra protocol packets inserted into the network by reliable diffusion worsen the congestion and thus lead to an increase in end-to-end delay.

We now compare the performance of RMST [24] to RDD and DD using the same three metrics (delivery ratio, energy/packet, and end-to-end delay). RMST provides a reliable transport layer. The designers of RMST conclude that the best implementation for reliability in distributed sensor network architectures involves both the transport and MAC layers[24]. In our simulations we ran RMST by operating each node in caching mode. The transport layer based NACK mechanism of RMST adds significant overhead to sensor networks. In Figure 8.26 it can be seen that the best delivery ratio can be achieved only when the data rate is about 1 PPS (about 90%). However, the energy consumed is about 10 times the energy consumed by either reliable diffusion or directed diffusion. The delivery ratio drops steadily as the data rate increases to 7 PPS (about 30%). It should be observed that these low data rates do not support video applications.

Figures 8.27, 8.29, and 8.31 show the performance of RMST for the random topology. Similar observations can be drawn for random scenario. From Figures 8.28 and 8.29 it can be seen that RMST consumes about 10 times more energy when the data rate is 7 PPS when compared to either standard diffusion or reliable diffusion. Even when the data rate is 1 PPS RMST still consumes about twice as much energy as RDD or directed diffusion. Figure 8.31 and 8.30 show that RMST has large end-to-end delays when the data rate is increased to 4 PPS. Under 4 PPS its end-to-end delay is comparable to that of both RDD and directed diffusion. Hence, it can be concluded that RMST is not feasible for high data rate sensor network applications such as video sensor networks.

Finally, Figure 8.39 demonstrates the scalability of reliable diffusion. Protocol overhead is measured as path length is increased from 10 hops to 80 hops. The grid topology was used and the layout was so chosen that there are a maximum of 3 backup nodes and a minimum of 1 backup node. The y-axis represents the protocol overhead generated by reliable diffusion (measured in packets) to identify the backup nodes at each link. The graph compares the theoretical protocol overhead estimated using Equation 6.3 to the simulated overhead. The protocol overhead deviates from the estimated values because the number of backup nodes identified at each link varies. To compute the theoretical value, we set the value of N (number of backup nodes at each link) in Equation 6.3 to 2 to match the value observed in the simulations. These results clearly verify that protocol overhead for RDD is linear with respect to path length.

Figure 8.38 shows the time taken by reliable diffusion to repair permanent route failures (caused by node failures). These failures are recovered in approximately 1-3 seconds depending on the data rate. We do not make similar measurements for directed diffusion because, in directed diffusion, routes are not recovered until the next time the source sends out exploratory data packets. Assuming that a permanent route failure happens t seconds after the simulation starts and the source sends out exploratory data packets at every T seconds ($T = 60$ seconds in our implementation), directed diffusion takes $T - (t \ mod \ T)$ seconds to repair failed routes. In RDD, the route repair algorithm is initiated as soon as 3 data packets are rerouted through a backup node. In Figure 8.38, reliable directed diffusion is operating in 2-level hierarchical mode. By doing so, the RDD can repair routes without broadcasting data packets.

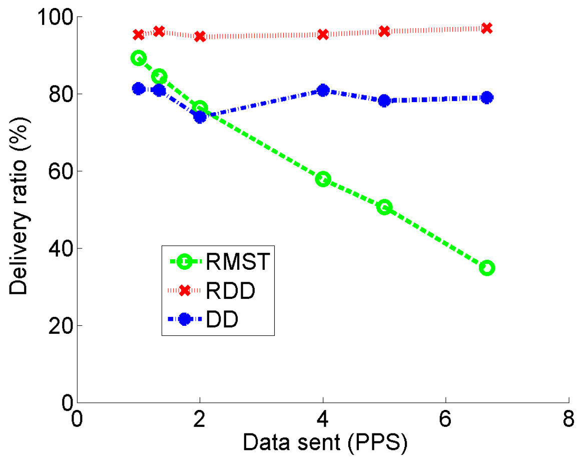

Figure 8.26: Delivery Ratio of RDD, RMST and DD for Grid topology.

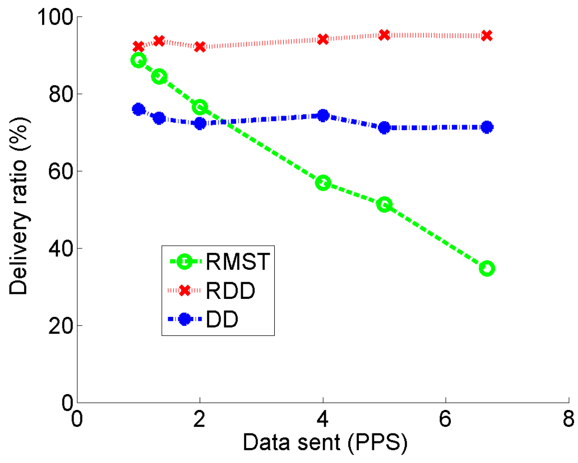

Figure 8.27: Delivery Ratio of RDD, RMST and DD for Random topology.

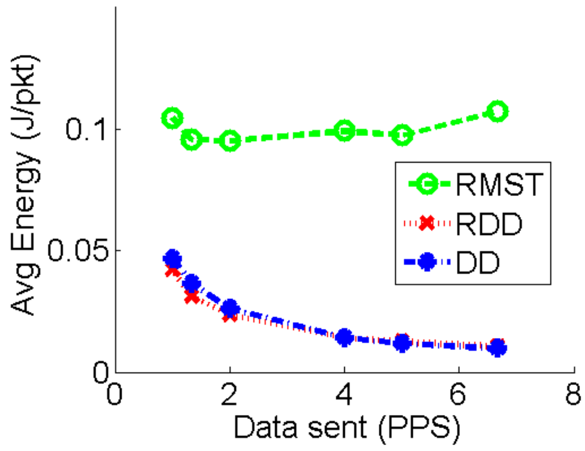

Figure 8.28: Average energy per packet consumed by RDD, RMST and DD Grid topology.

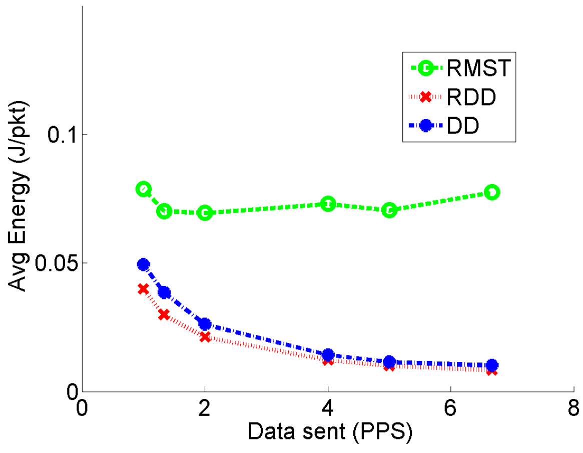

Figure 8.29: Average energy per packet consumed by RDD, RMST and DD Random topology.

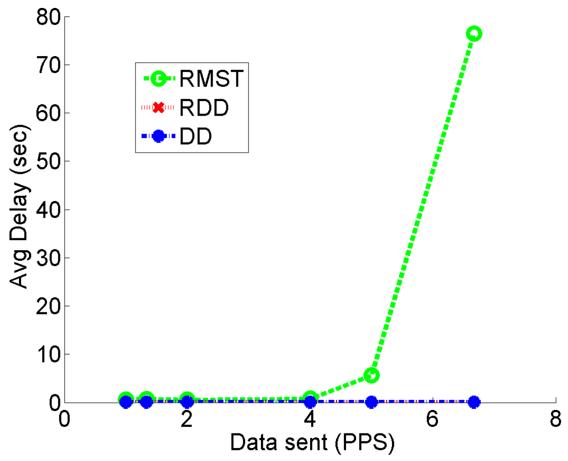

Figure 8.30: End-to-End Delay of RDD, RMST and DD for Grid topology.

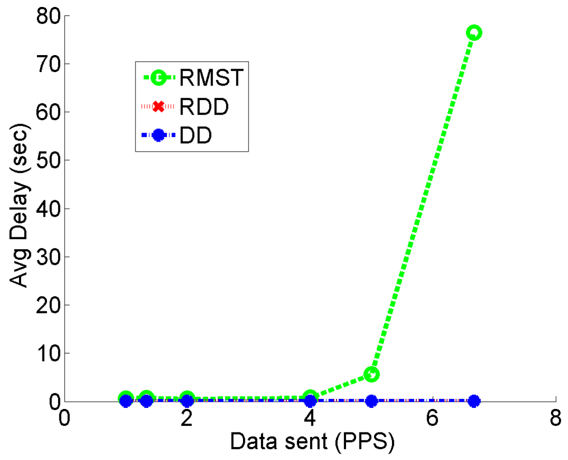

Figure 8.31: End-to-End Delay of RDD, RMST and DD for Random topology.

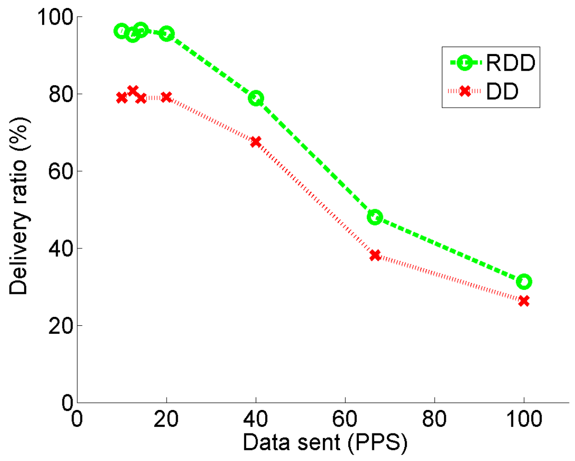

Figure 8.32: Delivery Ratio of RDD and DD for Grid topology

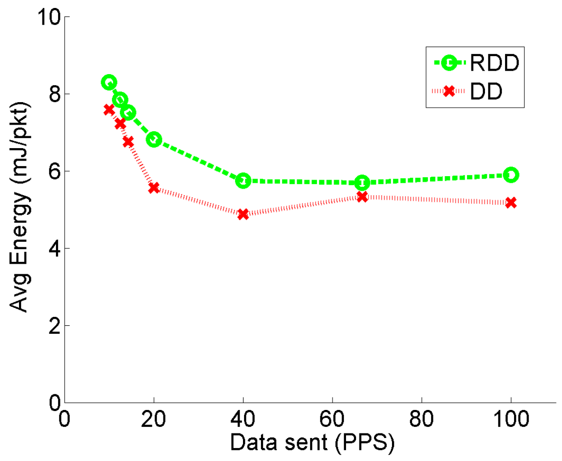

Figure 8.33: Average energy per packet consumed by RDD and DD for Grid topology

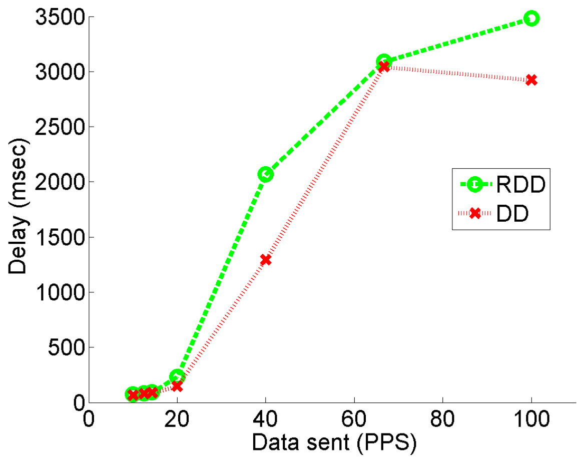

Figure 8.34: End-to-End Delay of RDD and DD for Grid topology

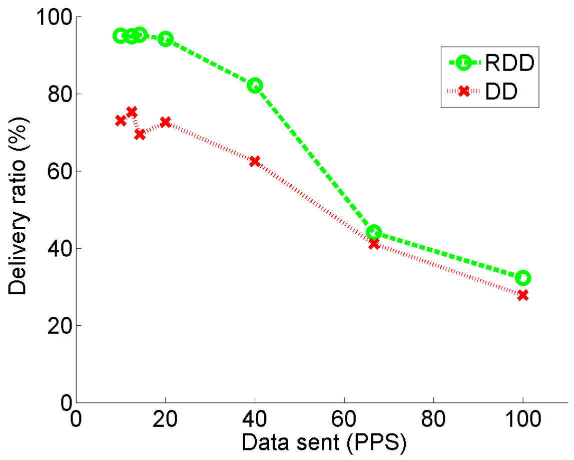

Figure 8.35: Delivery Ratio of RDD and DD for Random topology

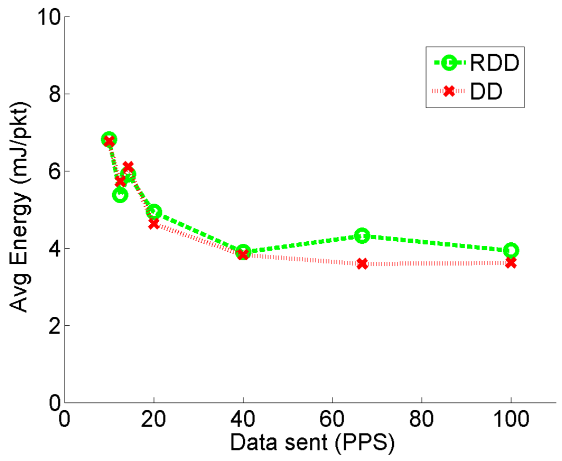

Figure 8.36: Average energy per packet consumed by RDD and DD for Random topology

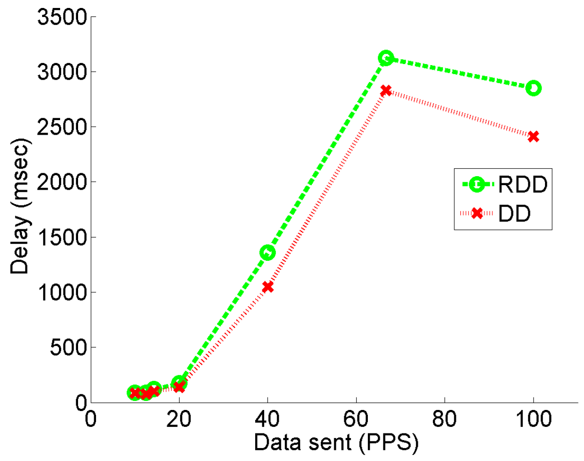

Figure 8.37: End-to-End Delay of RDD and DD for Random topology

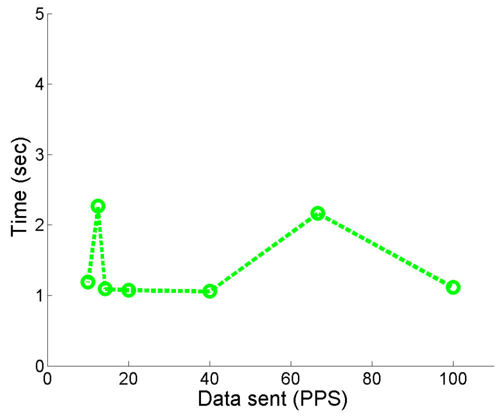

Figure 8.38: Time to repair routes due to node failures versus data send rate (grid topology)

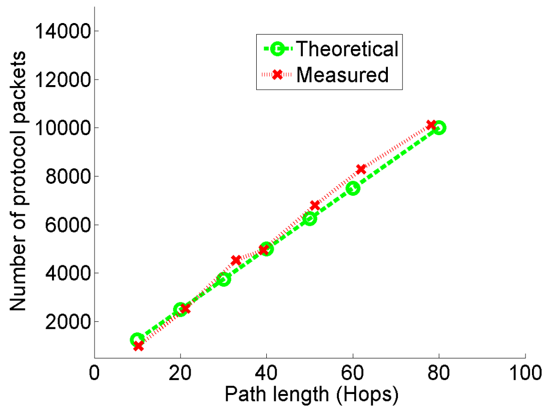

Figure 8.39: Protocol Overhead of RDD in case of Grid topology

Chapter 9
Conclusions and Future Work

In this work we have proposed a Robust Clustering Mechanism and Reliable Directed Diffusion mechanism to overcome the problems of the target tracking prototype that we have developed and make it a scalable application.

The proposed Robust Clustering Algorithm is a distributed clustering algorithm to track intruders. The localized nature of the algorithm ensures that reconfiguration does not significantly increase the protocol overhead. The algorithm builds a series of overlapping clusters which allow for more than one cluster to track a region. This redundancy improves the overall system reliability. The overlapping clusters also allow for tracking of curvilinear targets. We analyzed the factors that affect the protocol overhead. The protocol overhead depends on the sensor deployment strategy, number of clusters and the approach adopted by the cluster head in choosing its member nodes. Maximum coverage approach forms larger numbers of clusters than the minimum coverage approach and also leads to proportionate increase in the protocol overhead. Node failures lead to re-organization of clusters and hence increase protocol overhead. In the worst case there is a 11% increase in protocol overhead to reconfigure the clusters. This happens when the sensors are deployed in a grid.

The advantages of the proposed Robust Clustering Algorithm can be summarized as

- It is distributed in nature and the number of clusters to be formed can be easily controlled.
- A series of overlapping clusters improves system reliability and allows for tracking of curvilinear trajectories.

- Since the cluster head chooses its member nodes from its one hop neighbors, the raw data has to travel only one hop.

- Finally, the target tracking results of each cluster head can be progressively fused with those of its neighboring clusters.

We also proposed a Reliable Directed Diffusion, a protocol that can repair both temporary and permanent path failures. RDD makes use of the fact that nodes which are geographically closer to the sender experience better link quality than the nodes which are farther away. The results show that reliable diffusion improves delivery ratio by about 20% in the case of a grid topology and about 35% in a random topology. These improvements are achieved with just 6% increase in energy consumed or end-to-end delay. Furthermore, RDD has been shown to be a highly scalable protocol given that its overhead increases as a linear function of the path length. RDD significantly improves the reliability of directed diffusion for both permanent node failures and temporary link failures without incurring costly overhead. RDD make substantial contributions to directed diffusion and to data-centric routing.

The primary advantages of Reliable Directed Diffusion are:

- RDD provides high reliability and delivery ratio.

- RDD handles both link and node failures.

- RDD requires minimal configuration.

- Message overhead is linear with respect to path length and hence scalable to large networks.

So far we have been able to successfully simulate Target Tracking experiments in NS2. This has helped us to analyze various errors in the CPA algorithm and propose and test improvements to the algorithm quickly. While the reliable directed diffusion has been proposed to recover packets from link and node failures, its interaction with Robust Clustering Algorithm remains to be evaluated.

Th major difficulty in conducting Target Tracking experiments is deploying and configuring the network. Further it is prohibitively expensive to deploy 100's of sensors. Hence it is important to integrate both the proposed techniques into a single simulation environment that could be used to evaluate and answer questions like:

- Given a node deployment scenario what is the best possible cluster configuration?
- What is the optimum number of clusters to achieve a desired level of target detection?
- What is the number of additional sensors needed to further improve the target detection?
- How to understand the impact of node deployment on target detection accuracy and reliability in delivery of video packets from the clusterhead to external agents.

Bibliography

[1] F.M. Dommermuth. The estimation of target motion parameters from cpa time measurements in a field of acoustic sensors. In *J. Acoust. Soc. Am.*, volume 83, pages 1476–1480, 1988.

[2] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *Networking, IEEE/ACM Transactions on*, 11(1):2–16, Feb. 2003.

[3] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, New York, NY, USA, 2003. ACM.

[4] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001)*, 2001.

[5] R. Ogier. Topology dissemination based on reversepath forwarding (tbrpf, 2004.

[6] C. Santivanez and R. Ramanathan. Hazy sighted link state routing: A scaleable link state algorithm. *BBN Tech. Memo No.1301, Aug*, 31, 2001.

[7] C. Perkins. Ad hoc on demand distance vector (aodv) routing, 1997.

[8] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[9] Z. Haas and M. Pearlman. *Determining the Optimal Configuration for the Zone Routing Protocol, IEEE JSAC, Special Issue on Ad-Hoc Networks.* August 1999.

[10] Sung Ju Lee, William Su, and Mario Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *Mob. Netw. Appl.*, 7(6):441–453, 2002.

[11] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 207–218, New York, NY, USA, 1999. ACM.

[12] P. Sinha, R. Sivakumar, and V. Bharghavan. Mcedar: Multicast core extraction distributed ad-hoc routing. In *In Proceedings of the Wireless Communications and Networking Conference, WCNC 99.*, pages 1313–1317. September 1999.

[13] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Mobicom 99*, pages 207–218. August 1999.

[14] C.-K. Toh, G. Guichal, and S. Bunchua. Abam: on-demand associativity-based multicast routing for ad hoc mobile networks. *Vehicular Technology Conference, 2000. IEEE VTS-Fall VTC 2000. 52nd*, 3:987–993 vol.3, 2000.

[15] Lusheng Ji and M.S. Corson. A lightweight adaptive multicast algorithm. *Global Telecommunications Conference, 1998. GLOBECOM 98. The Bridge to Global Integration. IEEE*, 2:1036–1042 vol.2, 1998.

[16] L. Ji and M. S. Corson. Differential destination multicast (ddm), 2000.

[17] J. J. Garcia-Luna-Aceves and Ewerton L. Madruga. A multicast routing protocol for ad-hoc networks. In *INFOCOM*, pages 784–792, 1999.

[18] Bommaiah, McAuley, , and Talpade. Amroute: Adhoc multicast routing protocol, 1999.

[19] Ching-Chuan Chiang, Mario Gerla, and Lixia Zhang. Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks. *Cluster Computing*, 1(2):187–196, 1998.

[20] Jorjeta G. Jetcheva, Yih-Chun Hu, David A. Maltz, and David B.Johnson. A simple protocol for multicast and broadcast in mobile adhoc networks, 2001.

[21] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Mobile Computing and Networking*, pages 207–218, 1999.

[22] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. Psfq: a reliable transport protocol for wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 1–11, New York, NY, USA, 2002. ACM.

[23] Özgür B. Akan and Ian F. Akyildiz. Event-to-sink reliable transport in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 13(5):1003–1016, 2005.

[24] F. Stann and J. Heidemann. Rmst: reliable data transport in sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 102–112, 11 May 2003.

[25] Q. Cao, T. Abdelzaher, T. He, and R. Kravets. Cluster-based forwarding for reliable end-to-end delivery in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1928–1936, Anchorage, AK, USA,, May 2007.

[26] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 266–279, New York, NY, USA, 2003. ACM.

[27] Jorjeta G. Jetcheva and David B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 33–44, New York, NY, USA, 2001. ACM.

[28] D. Tian and N.D. Georganas. Energy efficient routing with guaranteed delivery in wireless sensor networks. *IEEE Wireless Communications and Networking*, 3:1923–1929, 2003.

[29] George Aggelou and Rahim Tafazolli. RDMAR: A bandwidth-efficient routing protocol for mobile ad hoc networks. In *WOWMOM*, pages 26–33, 1999.

[30] C. Toh. Associativity-based routing for ad-hoc mobile networks.

[31] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks, 1999.

[32] Ionut D. Aron and Sandeep K. S. Gupta. A witness-aided routing protocol for mobile ad-hoc networks with unidirectional links. In *MDA '99: Proceedings of the First International Conference on Mobile Data Access*, pages 24–33, London, UK, 1999. Springer-Verlag.

[33] D. Cerpa, A.; Estrin. Ascent: adaptive self-configuring sensor networks topologies. *Transactions on Mobile Computing*, 3(3):272–285, July-Aug. 2004.

[34] X. Xu and S. Sahni. Approximation algorithms for sensor deployment. In *Innovations and Real-Time Applications of Distributed Sensor Network, 2006. Proceedings. 2nd International Symposium on*, 2006.

[35] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IEEE IPSN*, 2004.

[36] T. Clouqueur, V. Phipatanasuphorn, Saluja K., and P. Ramanathan. Sensor deployment strategy for detection of targets traversing a region. In *Mobile Networks and Applications*, volume 28, pages 453–461, 2003.

[37] S. Kumar, T.H. Lai, and Balogh J. On k-coverage in a mostly sleeping sensor network. In *Mobile Computing and Networking, 2004. Proceedings. Tenth Annual International Conference on. ACM*, 2004.

[38] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *SenSys*, 2003.

[39] S. Megerian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Worst- and best-case coverage in sensor networks. In *Mobile Computing, 2004. IEEE Transactions on*, volume 3. IEEE, 2004.

[40] J. Adriaens, S. Megerian, and M. Potkonjak. Optimal worst-case coverage of directional field-of-view sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2006. The third annual IEEE Communications Society Conference on*, pages 336–345. IEEE, 2006.

[41] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. volume 46, pages 605–634. Computer Networks, 2004.

[42] H. Tian, A. Pascal, Y. Ting, L. Liqian, Z. Gang, S. Radu, C. Qing, A. Stankovic, and T. Abdelzaher. Achieving real-time target tracking using wireless sensor networks. In *Embedded Computing System, 2007. ACM Transactions on*, 2007.

[43] W. Qixin, C. Wei-Peng, Z. Rong, L. Kihwal, and S. Lui. Acoustic target tracking using tiny wireless sensor devices. In *Information Processing in Sensor Networks, 2003. Proceedings. 2nd International Workshop on*, 2003.

[44] W. Xiaoling and Q. Hairong. Mobile agent based progressive multiple target detection in sensor networks. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. IEEE International Conference on*, 2004.

[45] S. Oh, P. Chen, M. Manzo, and S. Sastry. Instrumenting wireless sensor networks for real-time surveillance. In *Robotics and Automation, 2006. Proceedings. the International Conference on*, 2006.

[46] Qing Yang, Alvin Lim, Kenan Casey, and Raghu Neelisetti. An enhanced cpa algorithm for real-time target tracking in wireless sensor networks. *International Journal of Distributed Sensor Networks.*

[47] Qing Yang, Alvin Lim, Kenan Casey, and Raghu Kisore Neelisetti. An empirical study on real-time target tracking with enhanced cpa algorithm in wireless sensor networks. *Ad Hoc and Sensor Wireless Networks An International Journal.*

[48] Qing Yang, Alvin Lim, Kenan Casey, and Raghu Kisore Neelisetti. Real-time target tracking with cpa algorithm in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*, pages 305–313, 2008.

[49] Raghu Kisore Neelisetti, Alvin Lim, Prathima Agrawal, and Qing Yang. A robust clustering algorithm for target tracking in wireless acoustic sensor networks. In *IRA-DSN 2007: Innovations and Real-time Applications of Distributed Sensor Networks (DSN) Symposium 2007*, pages 17–25, 2007.

[50] S. Li, R. Neelisetti, C. Liu, and A. Lim. Real-time target tracking with cpa algorithm in wireless sensor networks. In *a World of Wireless, Mobile and Multimedia Networks, 2008. WOWMOM '08. 9th IEEE International Symposium on*, 2008.

[51] K. Casey, R. Neelisetti, and A. Lim. Rtdd: A real-time communication protocol for directed diffusion. *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 2852–2857, 31 2008-April 3 2008.

[52] F. Silva, J. Heidemann, and R. Govindan. Network routing application programmer's interface, 2002.

[53] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.

[54] Weilian Su and Tat L. Lim. Cross-layer design and optimization for wireless sensor networks. *snpd-sawn*, 0:278–284, 2006.

[55] A. M. Safwat. A novel framework for cross-layer design in wireless ad hoc and sensor networks. In *GlobeCom Workshops 2004. IEEE*, pages 130–135, November/December 2004.

[56] T. Melodia, M. C. Vuran, and D. Pompili. The state-of-the-art in cross-layer design for wireless sensor networks. In *Springer Lecture Notes in Computer Science (LNCS)*, volume 3883, June 2006.

[57] Network simulator 2 (ns2), 1997.

[58] Branko Grunbaum and G.C. Sheperd. *Tilings and Patterns.* W.H Freeman And company, New York, U.S.A, 1986.

[59] Kishor S. and Trivedi. *Probability and statistics with reliability, queuing and computer science applications.* John Wiley and Sons Ltd., 2002.

[60] Wu Xiuchao and A. L. Ananda. Link characteristics estimation for ieee 802.11 dcf based wlan. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pages 302–309, Washington, DC, USA, 2004. IEEE Computer Society.

[61] Theodore Rapport. *Wireless Communications: Principles and Practice.* Prentice Hall, 2nd edition, 2001.